Radboud University
Faculty of Social Sciences
Artificial Intelligence

Master thesis

Fast Representatives Search as an Initialization for Scalable Sparse Subspace Clustering

J.P. Marsman
pietermarsman@gmail.com

supervised by
Dr. Jason Farquhar

August, 2015

Radboud University

Contents

List of Figures

Abstract

High-dimensional data clustering is a difficult task due to the sparsity, correlated features and specific subspace structures of high-dimensional data. However, the self-expressiveness property states that data points can be most efficiently represented as data points from their own subspace. This property is successfully used by Elhamifar and Vidal (2013) to cluster high-dimensional data with the sparse subspace clustering (SSC) algorithm. However, the computational complexity of SSC is too high to be applied on datasets with large number of data points.

Scalable sparse subspace clustering (SSSC) uses an in-sample out-of-sample approach to speed SSC up. Two steps were taken in this research to improve the random initialization of the in-sample set of SSSC. First, the computational complexity of an algorithm for representative selection called sparse representatives modeling selection (SMRS) was improved using a divide-and-conquer strategy. This new algorithm was called hierarchical sparse representatives (HSR). Secondly, the representatives from SMRS (or HSR) were used to initialize the in-sample set smartly.

Theoretical and empirical results indicated that SMRS and HSR had similar results. The representatives from both algorithms overlapped and the importance given to them correlated. However, using representatives or non-representatives as an initialization of the in-sample set of SSSC did not significantly change its performance.

Chapter 1

Introduction

Clustering is an unsupervised learning technique that imposes a group structure on data. This group structure is such that data points that are in the same group are more similar than data points that are in different groups (Aggarwal and Reddy, 2013).

Such a group structure can give data scientists new knowledge about the data. It describes what is considered as relatively similar or different. Also, the difference between the groups gives a notion about the underlying distribution or structure of the dataset. For this reason clustering is often used as a data exploration or feature generation tool.

However, clustering is often a daunting task because (a) multiple clusterings (e.g. an hierarchy of clusters) are possible, (b) cluster shapes are arbitrary, (c) there are many data points, (d) there are many features, (e) clusters are not well separated, (f) data points belong to multiple clusters, (g) outliers are present or (h) data is missing. Many algorithms solve a subset of these problems, but none of them solves them all.

The focus in this work is on clustering data with both a large number of features and a large number of data points. Algorithms that have a low complexity (i.e. compute a grouping relatively fast compared to the input size) are preferable in this setting.

With the rise of "big data" this type of data gets more common. Examples of high dimensional data (Kriegel et al., 2009) are DNA arrays, bag-of-words document representations, image (Tron and Vidal, 2007) or GPS trajectories and images (Lee et al., 2005). Often, the goal is to group multiple data points such that the data can be represented more efficiently or be better understood.

In the rest of this chapter the pitfalls of high-dimensional data clustering are explained. Also, some properties of high-dimensional data and state-of-the-art algorithms to cluster high-dimensional data are described. In the next chapter the used algorithms, datasets and measures in the experiments for this research are

described. Chapter 3 uses the knowledge from the first chapter to show the equivalence between the proposed and state-of-the-art algorithm. Also their complexity will be discussed. After that, the emperical results are described. Finally, Chapter 5 discusses the results and wraps them up in the conclusions.

1.1 High-dimensional data

Data with a lot of features (e.g. > 20) is called high-dimensional data. Compared to low-dimensional data it has several counter intuitive characteristics. In Section 1.1.1 these seemingly negative effects, also known as the curse of dimensionality, are discussed. After that two properties that we will use later on are explained in Section 1.1.2. With this knowledge we are ready to look at the algorithms that are used to cluster high-dimensional data.

1.1.1 Curse of dimensionality

The *curse of dimensionality* refers to the phenomenon that increased numbers of features make it harder to do something sensible with data. More precisely, the relative difference between data points that are "near" and data points that are "far away" becomes less when adding features. This is a general phenomena that occurs for each type of distance or similarity measurement. For examples, see Figures A.1-A.2 and also note that all the figures about the curse of dimensionality are in Appendix A. More formally (Aggarwal et al., 2001) for distance measure $d(x, y)$ between data points x and y with dimension D:

$$\lim_{D \to \text{inf}} \frac{\max(d(x,y)) - \min(d(x,y))}{\min(d(x,y))} = 0 \tag{1.1}$$

This is also illustrated by the D dimensional sphere with radius r described by Bishop (2006). The proportion of the volume of this sphere between $r = 1$ and $r = 1 - \epsilon$ is:

$$\frac{V_D(1) - V_D(1-\epsilon)}{V_D(1)} = 1 - (1-\epsilon)^D \tag{1.2}$$

For large D's this goes to one fast, meaning that all the volume for a high-dimensional sphere is in a tiny shell near the surface.

Consequently, the accustomed notion of distance in low-dimensional data, e.g. the straight line distance, becomes useless in high-dimensional data. At least three properties of most datasets are at the heart of this problem (Aggarwal and Reddy, 2013).

Firstly, in high-dimensional datasets the available data becomes sparse. For example, to represent a one-dimensional dataspace with twenty possible discrete

values, at least twenty data points are needed (also see Figure A.3). If an additional feature is added with also twenty possible values, in total 400 data points are needed to represent all the combinations (also see Figure A.4). Hence, an exponential number of data points in the dimension is needed to densely fill a dataspace. Consequently, with the same number of data points to represent the whole dataspace, the data becomes sparse and concepts of distance and similarity matter less.

Secondly, features are often correlated and thus the data can actually be represented in a lower dimension (e.g. compare Figure A.5 with Figure A.3). An example of such data are pixels of an image that are correlated with their neighbouring pixels. These features have redundant information that do not provide new information about the underlying structure of the data, but (possibly) do introduce new noise. Removing this correlation between the data, e.g. using principal component analysis, recovers the underlying structure. Consequently, for most high-dimensional datasets this underlying structure has a much lower dimension. For example, a subject from the Extended Yale B dataset can be represented in nine dimensions instead of 32256 pixels (Basri and Jacobs, 2003). The image trajectories of the Hopkins 155 datasets can be represented in four dimensions (Tomasi and Kanade, 1992).

Thirdly, this redundancy or covariance between features often differs for groups of data points. These groups of points are so-called subspaces. This means that they only vary in a subset of the whole dataspace, e.g. into a particular direction. Because these subspaces are differently oriented for different groups of data points, distance measures that are sensitive to the variation in the data as a whole become less useful.

The sparsity, correlated features and subspace structure of high-dimensional data make it hard to use spatial relations. However, the subspace structure gives rise to a new kind of relation between data points. This self-expressiveness property is discussed in the next section.

1.1.2 Self-expressiveness property of subspaces

A subspace is a restricted part of the whole dataspace. A linear subspace is a subset of the whole dataspace that is closed under addition and multiplication. For example, both the lines and the plane in Figure 1.1 are subspaces of the three-dimensional dataspace.

In the following sections the concepts of subspaces (Section 1.1.2), the boundary of subspaces, i.e. convex hulls (Section 1.1.2) and lastly the self-expressiveness property (Section 1.1.2) are explained.

Subspaces

High-dimensional data is often structured as a union of (linear) subspaces. A linear subspace $\mathcal{S}_i$ of a D-dimensional space $\mathcal{S}$ is a d_i dimensional space, where $d_i < D$. A projection matrix $\mathbf{R}_i \in \mathbb{R}^{d \times D}$ relates data points in $\mathcal{S}_i$ to $\mathcal{S}$: $\mathbf{Y} = \mathbf{X}_i\mathbf{R}_i$, where $\mathbf{Y} \in \mathcal{S}$ and $\mathbf{X}_i \in \mathcal{S}_i$. Because $\mathbf{Y}$ is just the projected data $\mathbf{X}_i$ it has the same low-rank d_i but it has dimension D.

Let $\{\mathcal{S}_i\}_{i=1}^n$ be a union of n subspaces. A dataset $\mathbf{Y}$ with N data points from this union of subspaces is:

$$\mathbf{Y} \triangleq [\mathbf{y}_1 \dots \mathbf{y}_N] = [\mathbf{Y}_1 \dots \mathbf{Y}_n]\Gamma \tag{1.3}$$

where $\mathbf{Y}_i \in \mathbb{R}^{D \times N_i}$ is sampled from $\mathcal{S}_i$ and has rank d_i. The data points are permutated by Γ and therefore it is not known which data points belong to which subspace.

Convex hulls

Data points that are most extreme into a particular direction are data points that lie on the so-called convex hull. Intuitively, one can think of the convex hull as a rubber band that is stretched around all the data points. Once it is released, it contracts until it gets stuck on the most outer points of the dataset (also see Figure 1.2).

More formally the convex hull of a finite set is the region or set of data points that can be expressed as a linear combination of all the data points, where the linear combination has non-negative components that sum to one (de Berg et al., 2008):

$$\left\{ \sum_{i=1}^{|S|} \alpha_i x_i \quad \middle| \quad \sum_{i=1}^{|S|} \alpha_i = 1 \wedge \forall_i \alpha_i > 0 \right\} \tag{1.4}$$

In reality, the set of data points $\{x_i\}$ that can only be represented using itself as a linear combination (only using $\alpha_i = 1$) are meant when talking about the convex hull. The other data points that can be expresses as a linear combination of others are the ones that are inside the convex hull.

Convex hulls for high-dimensional datasets are hard to compute, e.g. the quick-hull algorithm takes at least $\mathcal{O}(ND^2)$ (Barber et al., 1996). Furthermore, the proportion of data points that is on the convex hull of an arbitrarily distributed dataset becomes exponentially closer to one when the number of features grows linearly. For example, the volume of a sphere that is close to the surface increases exponentially with the number of features (see Section 1.1.1).

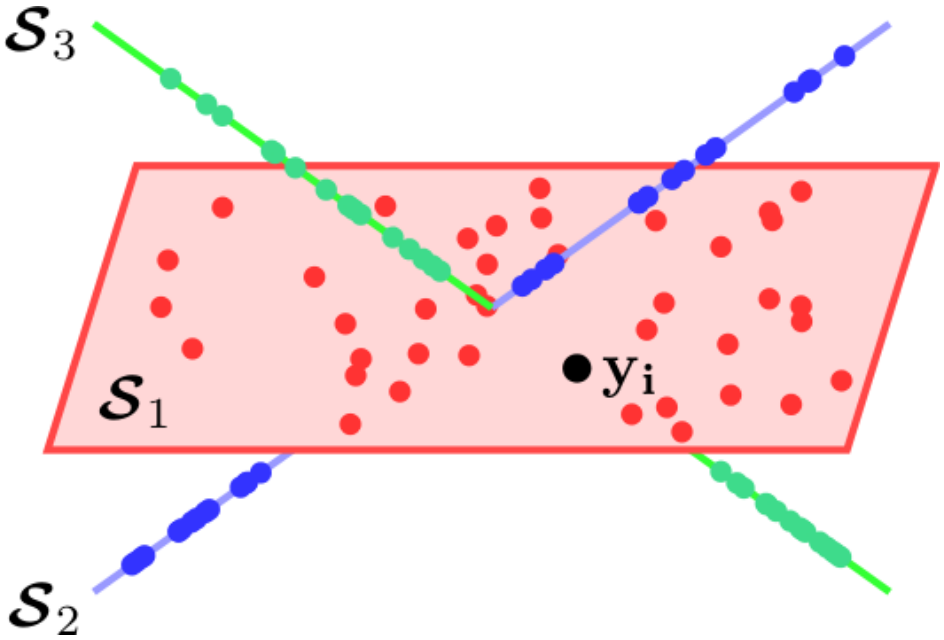

Figure 1.1: Three subspaces of a three-dimensional dataspace. Originally from Elhamifar and Vidal (2013).

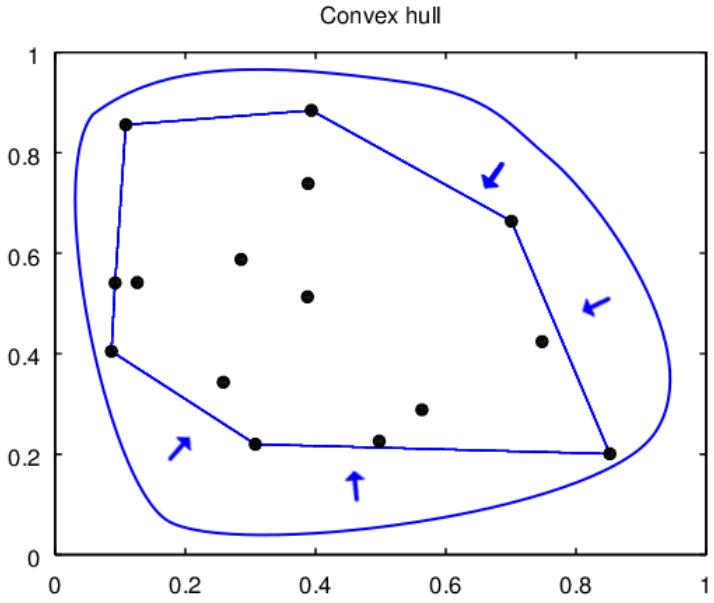

Figure 1.2: The convex hull can be thought of as a rubber band (or its multi-dimensional equivalent) which is stretched around all the data points. Once it is released it keeps contracting until it gets stuck on the most outer data points.

Self-expressiveness property

In general, every D-dimensional data point can be represented as a linear combination of D linearly independent data points (Lay, 2012). Linearly independent means that each of the data points v_i can not be represented as a linear combination of the other data points, or more formally:

$$\text{if} \quad a_1 v_1 + a_2 v_2 + \cdots + a_k v_k = 0 \quad \text{then} \quad a_1 = a_2 = \cdots = a_k = 0 \tag{1.5}$$

Now imagine a set of data points in a D-dimensional space that are actually in an d-dimensional subspace. These data points can be represented as a linear combination using only d linearly independent data points instead of D linearly independent data points. More formally the self-expressiveness property says that data points in a linear subspace can efficiently be represented as a linear combination of other data points in the same subspace. In this case, efficiently means that only d data points should be used instead of D data points and likely $d << D$.

In reality this means that when representing data points as a linear combination of others, only d points from the same subspace have to be used. When using data points of other subspaces, D data points need to be used. This means that the data points in the same cluster do not have to be close to each other but should be easily represented as one another. For this exact reason, the data point y_i in Figure 1.1 intuitively belongs to the red class while it is closer to many of the green and blue data points. This property is used in sparse subspace clustering (SSC) as a indication for which data points are in the same subspace or cluster.

1.2 Traditional clustering approaches

Traditional clustering approaches are based on the (relative) distance or similarity between data points [1]. The assumption is that data points that are far away or not similar are less likely to belong to the same cluster than data points that are close and similar. Traditionally the aim of clustering is thus dividing the data points into groups with minimal within-group distances and maximal between-group distances (Aggarwal and Reddy, 2013).

Different interpretations of this goal lead to different kind of traditional clustering algorithms. Centroid-based clustering algorithms interpret the within-group distances as the distance to the centroid and the between-group distance as the distance between the centroids. Distribution-based clustering algorithms additionally differentiate in which direction the distance is measured. Density-based

[1] In theory, it does not matter if we talk about distance or similarity and thus we will use the terms interchangeably. In reality however these differences are harder to overcome since positive measures are sometimes preferred.

clustering algorithms group dense regions and consequently grouping data points with low distances. Finally, connectivity-based clustering algorithms group two or more data points that are similar. These algorithms are all susceptible for the pitfalls of high dimensional data explained in the previous section.

But as we have seen earlier, not the distance between two high-dimensional data points is important but the distance between these data point and the basis (or manifold) of the cluster it belongs to. This is illustrated nicely by the toy problem in Figure 1.1, but also true for actual high-dimensional datasets. For example, many data points from a subject in the Extended Yale B dataset are nearer to data points from other subjects than to data points from the same subject (Elhamifar and Vidal, 2013) (also see Figure 1.3). For this reason, neither of the traditional clustering algorithms clusters the Extended Yale B data correctly because they are based on the distance measures between data points which do not reveal the underlying structure.

A solution would be to incorporate knowledge of the linear basis of each cluster into the distance measure. However, the linear basis for each cluster is unknown. The next section treats this subject of simultaneously finding linear bases and grouping the data points.

1.3 Subspace clustering

Algorithms that are specifically designed for high-dimensional data clustering are often called subspace clustering algorithms. Clustering subspaces is easy when having the basis for each subspace, just assign each data point to the closes subspace. Also, when having the clustering of the data it is easy to obtain a basis for each group. However, both the clustering and the basis for each subspace are unknown and thus the goal of subspace clustering is twofold (Vidal, 2011): (a) assign each data point to one of two or more groups and (b) estimate the linear basis for each of the groups.

Simultaneously finding a basis and clustering the dataset is a chicken-and-egg problem. Without a proper basis for each group, data points cannot be assigned to a group. Without a clustering of the dataset, the linear basis for each group cannot be computed.

Several subspace clustering algorithms are described by Vidal (2011) in his excellent review. *Algebraic methods* factorize the data matrix into several low-rank components. The low-rank components can be computed simply from an SVD (Costeira and Kanade, 1998) or more sophisticatedly using higher-degree polynomials (Vidal et al., 2005). *Iterative methods* switch between assigning each data point to a cluster and computing the linear basis for each cluster using principal components analysis (PCA) (Mustafa, 2004). It is similar to iterative methods for

low-dimensional data but instead of centroids it uses subspaces. Other algorithms are more *statistically* and try to fit a probabilistic PCA or use Random Sample Consensus (RANSAC) (Fischler and Bolles, 1981) to find multiple linear bases. Another statistical method is agglomerative lossy compression (ALC) (Derksen et al., 2007) that repetitively fuses two groups that decreases the coding length the most.

A last set of algorithms is based on *spectral clustering* (Luxburg, 2007) of an affinity matrix. The core difference between algorithms in this category is which affinity matrix they use. Affinity matrices can be based on the (Gaussian, angular, etc.) similarity between data points, the output of other algorithms or the coefficient for writing a data point as a linear combination of others (Vidal, 2011). Low-rank representation (LRR) (Chen and Yang, 2014) minimizes the rank of the linear representation of all the data points as each other. Sparse subspace clustering (SSC) (Elhamifar and Vidal, 2009, 2010) uses a sparse minimization of a linear representation of data points as each other.

Currently, SSC is one of the best subspace clustering algorithms. The next section (Section 1.3.1) explains how SSC works and why it is slow. Section 1.3.2 describes a fast approximation of SSC called scalable sparse subspace clustering (SSSC). Then a variation on SSC to find representatives is explained (Section 1.3.3). In the rest of this research these representatives are used to initialize SSSC to get a better clustering performance.

1.3.1 Sparse subspace clustering

The self-expressiveness property (Section 1.1.2) suggests that if a data point is linearly represented as little data points as possible, these data points lie in the same subspace. Thus minimizing the number of data points used for each reconstruction, i.e. the ℓ_0-norm, gives a strong indication on which data points belong in the same subspace and thus same cluster. However, an ℓ_0 minimization is a combinatorial problem and thus NP-hard.

Sparse subspace clustering, created by Elhamifar and Vidal (2013) (Algorithm 1), uses a ℓ_1-norm instead of a ℓ_0 norm. The ℓ_1-norm is the tightest convex relaxation of the ℓ_0-norm and is known to have similar sparse solutions (Donoho, 2006). Consequently, the sparse results can be computed efficiently.

Simultaneously minimizing the reconstruction error (in case of noisy data) and minimizing the sum of the coefficients (the ℓ_1-norm) is known as least absolute shrinkage and selection operator or lasso. Lasso can be computed using convex optimization and thus finding minimal coefficients $\mathbf{c}_i^*$ for each data point $\mathbf{y}_i$ becomes feasible:

Figure 1.3: Two images from the Extended Yale B dataset that have a very high cosine similarity (0.9782) but are not from the same subset.

Algorithm 1 Sparse subspace clustering

Input: A set of points $\{\mathbf{y}_i\}_{i=1}^N$ lying in a union of n linear subspaces $\{\mathcal{S}_i\}_{i=1}^n$

1. Solve the sparse optimization program in Equation 1.6.
2. Normalize the columns of $\mathbf{C}$ as $\mathbf{c}_i \leftarrow \frac{\mathbf{c}_i}{\|\mathbf{c}_i\|_\infty}$
3. Form a similarity graph with N nodes representing the data points. Set the weights on the edges between the nodes by $\mathbf{W} = |\mathbf{C}| + |\mathbf{C}|^\top$
4. Apply spectral clustering to the similarity graph

Output: Segmentation of the data: $\mathbf{Y}_1, \mathbf{Y}_2, \ldots, \mathbf{Y}_n$,

$$\mathbf{c}_i^* = \operatorname{argmin} \|\mathbf{c}_i\|_1 + \frac{\alpha}{2} \|\mathbf{y}_i - \mathbf{Y}\mathbf{c}_i\|_F^2 \quad \text{s.t.} \quad \mathbf{c}_{ii} = 0 \tag{1.6}$$

Together the coefficients $\mathbf{c}_i^*$ create a square matrix $\mathbf{C}$ that defines how each data point is expressed as a linear combination of others ($\mathbf{Y} = \mathbf{YC}$). Before using this coefficient matrix as an affinity matrix in spectral clustering it is adapted in the following way. To be invariant of the norm of the data point, the coefficients are normalized by $\mathbf{c}_i^* \leftarrow \mathbf{c}_i^* / \|\mathbf{c}_i^*\|_\infty$ such that the affinity matrix is not dominated by the data points that are furthest from the origin. Then the affinity matrix $\mathbf{W}$ is constructed by making the normalized coefficients symmetrical $\mathbf{W} = |\mathbf{C}| + |\mathbf{C}|^\top$.

The coefficients $\mathbf{C}$ can be used as a affinity matrix because $\mathbf{c}_i^*$ has non-zero components for data points that are both close and in the same subspace. Consequently, $\mathbf{C}$ forms the weighting of a connectivity graph that connects each data point to other near data points in the same subspace. Thus the coefficients are a subspace specific similarity measure.

Ideally the similarity matrix is block sparse (see Figure 1.4), i.e. it has only non-zero coefficients for data points in the same subspace. Block sparse matrices can be clustered efficiently using spectral clustering.

Spectral clustering (Luxburg, 2007) considers the similarity matrix as a connectivity matrix in a weighted graph $\mathcal{G} = (\mathcal{V}, \mathcal{E}, \mathbf{W})$. Then the clustering is obtained by finding n different subsets of vertices's that are densely connected. First, the Laplacian of the similarity matrix is computed. The the first n eigenvectors are computed. The items of these eigenvectors are clustered using k-means.

Elhamifar and Vidal (2013) showed that SSC can indeed successfully be used for subspace clustering. Using the ℓ_1 norm results in sparse coefficients that can be divided into groups using spectral clustering. In fact, the prove that it always works when

$$\max_{\tilde{\mathbf{Y}}_i \in \mathbb{W}_i} \sigma_{d_i}(\tilde{\mathbf{Y}}_i) > \sqrt{d_i} \, \|\mathbf{Y}_{-i}\|_{1,2} \max_{j \neq i} \cos(\theta_{ij}) \tag{1.7}$$

where $\mathbb{W}_i$ is the set of all full-rank submatrixes $\tilde{\mathbf{Y}}_i \in \mathbb{R}^{D \times d_i}$ of $\mathbf{Y}_i$, $\sigma_{d_i}(\tilde{\mathbf{Y}}_i)$ is the d_i'th singular value of $\tilde{\mathbf{Y}}_i$, and $\cos(\theta_{ij})$ is the smallest principal angle between subspace $\mathcal{S}_1$ and $\mathcal{S}_2$. In other words, correct coefficients for subspace $\mathcal{S}_1$ are found if the smallest variation within the subspace (left hand side) is higher than the cosine similarity between the subspaces times some constant that is bases on the data (right hand side).

To verify this theoretical result in reality, Elhamifar and Vidal (2013) created linear subspaces with different cosine similarities and different number of data points per subspace. The results are shown in Figure 1.5. A similar result is obtained when varying the amounts of noise and the cosine similarities (see Figure C.1). Summarizing, when either the noise level or the cosine similarity is too high

or the number of data points per subspace is too low, SSC performs poorly. In the other cases it finds the subspace structure successfully.

The complexity of SSC is $\mathcal{O}(tN^3D)$, where t is the number of iterations used in the minimization, N the number of data points and D the number of features.

On real datasets SSC performs well. On average it has a 2.18% (median 0.00%) clustering error on the Hopkins-155 and 4.31% (median 2.50%) on the Extended Yale B dataset (5 subsets).

These results show that SSC is an algorithm that has state-of-the-art performance. However, SSC is not suitable for larger datasets because its complexity is cubic in the number of data points: $\mathcal{O}(tN^3D)$ (also see Section 3.2). The following section discusses a method that improves the complexity of SSC.

1.3.2 Scalable sparse subspace clustering

Scalable sparse subspace clustering (SSSC) (Algorithm 2) improves the complexity of SSC (Peng et al., 2013a). Instead of computing a linear representation for all data points over all data points, it uses a subset. This subset is clustered with SSC and then the model is used to cluster the other data points.

This is an out-of-sample approach, since part of the data is not in the sample that is used to build a first model. This out-of-sample data is treated as new data that arrives after the initial model building. To do this, SSSC takes a fixed number of data points from the dataset and applies SSC to it. At least d_i points from each subspace $\mathcal{S}_i$ are required to get a block sparse coefficient matrix.

Once the group labels for each data point are known the out-of-sample data is considered. A linear representation of all the out-of-sample data points over the in-sample data points is created using regularized linear regression. This second optimization has a much lower complexity than the default SSC optimization because only the in-sample data points are used as a dictionary and not all data points. Consequently the linear representation $\mathbf{C}^*$ of the out-of-sample data $\bar{\mathbf{Y}}$ over the in-sample data $\mathbf{X}$ can be relatively easily computed with:

$$\mathbf{C}^* = (\mathbf{X}^\top \mathbf{X} + \lambda \mathbf{I}) \cdot \mathbf{X}^\top \bar{\mathbf{Y}} \tag{1.8}$$

To assign the out-of-sample data to an actual group the reprojection error for each in-sample group is computed. The reprojection error is the euclidean distance between the actual data point and the reconstruction made by representing the data point as a linear combination of the in-sample data points in a single group. The size of the coefficients are also taken into account, i.e. lower coefficients are better.

Peng et al. (2013a) tested the influence of the in-sample data. Surprisingly, SSSC already performs pretty well using only very few in-sample data points. And

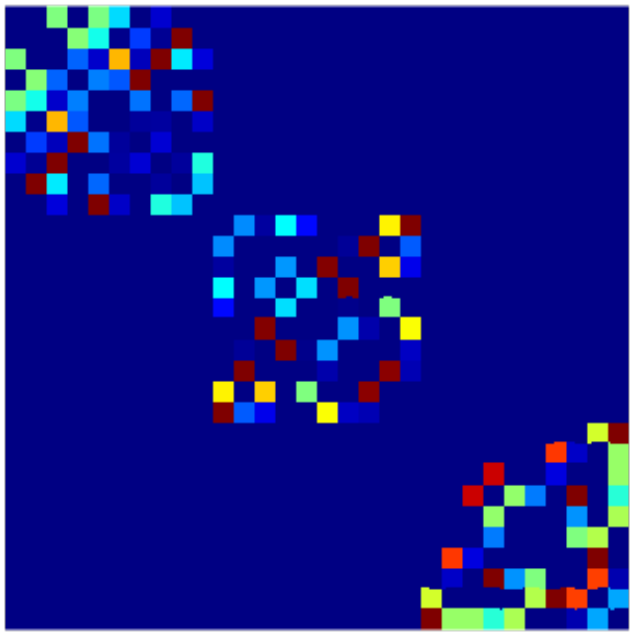

Figure 1.4: The block-sparse similarity matrix or connectivity matrix is created using $\mathbf{W} = \|\mathbf{C}\| + \|\mathbf{C}\|^{\top}$

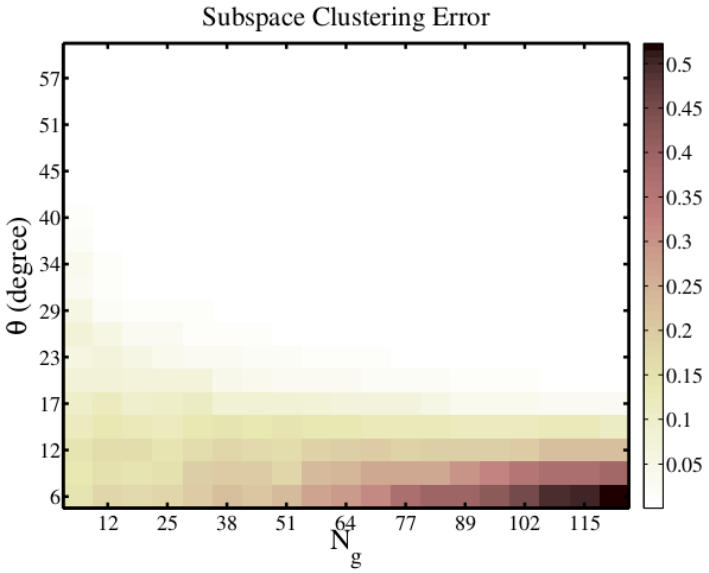

Figure 1.5: Clustering error for different numbers of data points per subspace and cosine angles (low degree is high similarity) between the subspaces. Clustering is not successful when either the number of data points per subspace is to low or cosine similarity is too high (Elhamifar and Vidal, 2013). Originally published by Elhamifar and Vidal (2013).

Algorithm 2 Scalable sparse subspace clustering

Input: A set of data points $\mathbf{Y} = \{\mathbf{y}_i\}_{i=1}^{N}$ from a union of n subspaces $\{\mathcal{S}_i\}_{i=1}^{n}$
Input: The ridge regression parameter λ

1. Select p data points from $\mathbf{Y}$ denoted by $\mathbf{X} = (x_1, x_2, ..., x_p)$.
2. Perform SSC over $\mathbf{X}$.
3. Calculate the linear representation of out-of-sample data $\bar{\mathbf{X}}$ over $\mathbf{X}$ by $\mathbf{C}_i^* = (\mathbf{X}^\top \mathbf{X} + \lambda \mathbf{I}) \cdot \mathbf{X}^\top \bar{\mathbf{X}}$
4. Calculate the normalized residuals of $\bar{x}^i \in \bar{\mathbf{X}}$ over all classes by
$$r_j(\bar{x}_i) = \frac{||\bar{\mathbf{x}}_i - \mathbf{X}\delta_j(\mathbf{c}_i^*)||_2}{||\delta_j(\mathbf{c}_i^*)||_2}$$
5. Assign $\bar{\mathbf{x}}_i$ to the class which produces the minimal residual by $\text{identity}(\bar{\mathbf{x}}_i) = \underset{j}{\operatorname{argmin}}\, r_j(\bar{\mathbf{x}}_i)$.

Output: Segmentation of the data: $\mathbf{Y}_1$, $\mathbf{Y}_2$, ..., $\mathbf{Y}_n$

when the in-sample size increases the performance increases as expected. Sadly, Peng et al. (2013a) did not compare the results of SSC and SSSC but compared it to other subspace clustering algorithms.

The simple technique of selecting only a few data points for the in-sample set improves the complexity of SSC enormously but also has an effect on the performance. Random selection is supported by the fact that with higher dimensions, more volume is close to the surface and thus no data point stands out as ideal for in-sample data. However, the actual structure of the subspaces has a much lower dimension (that is why SSC works) and thus in reality cleverly chosen points might be beneficial. In the next chapters we will research if representative or non-representatives can improve the SSSC performance.

1.3.3 Sparse modeling representatives selection

A variation on SSC called sparse modeling representatives selection (SMRS) is used for selecting representatives, also called examplers, from a dataset (see Algorithm 3). Representatives are a summary of the dataset, i.e. every data point is similar to one of the representatives. Representatives are useful for data discovery and data reduction.

Algorithm 3 Sparse modeling representatives selection

Input: A set of points $\{\mathbf{y}_i\}_{i=1}^N$ lying in a union of n linear subspaces $\{\mathcal{S}_i\}_{i=1}^n$

1. Solve the sparse optimization program in Equation 1.10.
2. Order the representatives: $\|\mathbf{c}_{i_1}\|_q > \|\mathbf{c}_{i_2}\|_q > \cdots > \|\mathbf{c}_{i_k}\|_q$

Output: Representatives: $i_1, i_2, \ldots, i_k$

Like SSC, SMRS also does a regularized minimization but instead of using the ℓ_1-norm it uses the row-sparsity promoting $\ell_{1,q}$-norm of the matrix $\mathbf{C}$. The $\ell_{1,q}$-norm is the sum over the ℓ_q-norm of the rows of the matrix:

$$\|\mathbf{C}\|_{1,q} \triangleq \sum_{i=1}^{N} \|\mathbf{c}_i\|_q \tag{1.9}$$

The optimization function thus becomes:

$$\mathbf{C}^* = \underset{\mathbf{C}}{\operatorname{argmin}}\, \alpha \|\mathbf{C}\|_{1,q} + \frac{1}{2}\|\mathbf{Y} - \mathbf{Y}\mathbf{C}\|_F^2 \quad \text{s.t.} \quad \mathbf{1}^\top \mathbf{C} = \mathbf{1}^\top \tag{1.10}$$

A typical resulting coefficient matrix from SMRS is shown in Figure 1.6. Besides the block sparsity that is promoted by the structure of the data, the optimization function also promotes row-sparsity in the coefficient matrix. Only data points that reduce the reprojection error significantly (depending on α) are non-zero.

Non-zero rows are representatives that are used to reconstruct multiple other data points. The representatives can be ordered on importance since rows that are being used in the reconstruction of more data points have a higher euclidean norm:

$$\|\mathbf{c}_{i_1}\|_q > \|\mathbf{c}_{i_2}\|_q > \cdots > \|\mathbf{c}_{i_k}\|_q \tag{1.11}$$

By definition, the data points on the convex hull of a subspace can represent all other data points inside the convex hull with coefficients that sum to one. Data points that are inside the convex hull need coefficients that sum to more than one to represent the data points that are on the convex hull. Because the data points on the convex hull of the subsets are most efficient in representing the whole subset the representatives found by SMRS are an approximation of the data points that are on the convex hull (Elhamifar et al., 2012). This theoretical result is used in Section 3.1 to show that the computations of SMRS can be split up without any losses.

In reality SMRS is indeed able to select representatives from complex data. Using video frames as data points, Elhamifar et al. (2012) showed that SMRS selected one or more representatives from each scene of the video.

The results from Elhamifar et al. (2012) show that SMRS is successful at selecting representatives. However, it's complexity is just as large as SSC: $\mathcal{O}(tN^3D)$. In the next sections we will discuss how SMRS can be computed faster using a divide-and-conquer strategy and we test if the representatives can be used to get better results from SSSC.

1.4 Our contribution

As we have seen in this chapter, SSC does a pretty good job at high-dimensional data clustering. Furthermore, SSSC does a pretty good job at improving the complexity of SSC. However, randomly selecting data points for the in-sample set is a little unsatisfactory and cleverly chosen points could improve the performance of SSSC.

Furthermore, SMRS does a pretty good job at selecting representatives from high-dimensional datasets. However, it is not applicable to large datasets due to the large complexity. Improving the complexity of SMRS and combining it with SSSC might be the way to go for easy interpretable and fast clustering of large high-dimensional datasets.

For these reasons the aim of this research is to devise an algorithm with reasonable complexity that computes representatives and clustering high-dimensional data accurately.

The first contribution of this research is an algorithm that uses a divide-and-conquer strategy to be able to compute representatives within reasonable time. This algorithm is called hierarchical sparse representatives. A second contribution is a proof of the equivalence between the SMRS and the new algorithm. A third contribution is the use of representatives or non-representatives as in-sample data for SSSC, and an experimental comparison of these approaches. The corresponding research questions are:

RQ 1. *Do sparse modeling representatives selection and hierarchical sparse representatives find the same representatives?*

RQ 2. *How can representatives be used for efficient high-dimensional data clustering?*

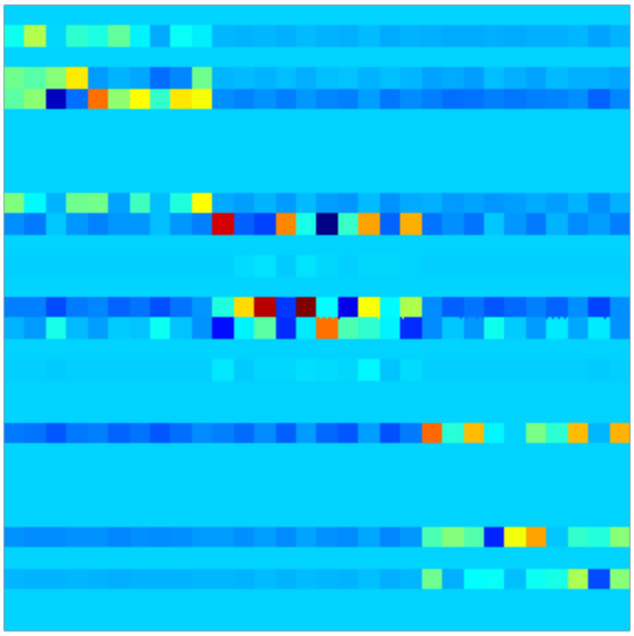

Figure 1.6: A row-sparse coefficient matrix created by SMRS

Chapter 2

Methods

Two new algorithms were introduced in this research. The first was called hierarchical sparse representatives (HSR) and used convex hull properties to compute representatives more efficiently, it is introduced in Section 2.1.1. The second algorithm was row-sparse subspace clustering (RSSC) and used information from SMRS (or HSR) to initialize the in-sample set of SSSC properly, it is introduced in Section 2.1.2.

To validate that SMRS and HSR have similar results, a theoretical proof based on convex hull properties was given. A second theoretical result was the complexity derivation for SMRS and HSR and their break-even point. Those findings are directly introduced in Chapter 3 and don't need any introduction in this methods chapter.

Multiple empirical comparisons were made between the new algorithms (HSR and RSSC) and the existing ones for representative finding (SMRS) and subspace clustering (SSC and SSSC). Three empirical experiments were used for this and described in detail in Section 2.4. The first experiments was used to determine the optimal parameters for all of the datasets that we test on. The second experiment compared the results of SMRS and HSR. The third experiment compared the clustering performances of SSC, SSSC and RSSC. The different used datasets are described in 2.2 and the two measures to compute the clustering performance in Section 2.3.

2.1 Algorithms

2.1.1 Hierarchical sparse representatives

The hierarchical sparse representatives algorithm splits the computation of SMRS into parts (see Algorithm 4). Instead of computing SMRS on the whole dataset,

Algorithm 4 Hierarchical sparse representatives

Input: A set of data points $\mathbf{Y} = \{\mathbf{y}_i\}_{i=1}^{N}$ from a union of n subspaces $\{\mathcal{S}_i\}_{i=1}^{n}$
Input: Maximum number of representatives N_{rep}
Input: Branching factor h
  $\mathbf{r}^{\text{out}} = \{1, 2, \ldots, N\}$
  while length($\mathbf{r}^{\text{out}}$) $> N_{\text{rep}}$ **do**
    Randomly divide the dataset $\mathbf{Y}$ into h parts: $\mathbf{Y}_1, \mathbf{Y}_2, \ldots, \mathbf{Y}_h$
    $\mathbf{r} = \text{SMRS}(\mathbf{Y}_j)$
    $\mathbf{r}^{\text{out}} = \{\mathbf{r}^{\text{out}}_{\mathbf{r}_i} \mid \mathbf{r}_i \in \mathbf{r}\}$
    $\mathbf{Y} = \{\mathbf{Y}_{\mathbf{r}_i} \mid \mathbf{r}_i \in \mathbf{r}\}$
  end while
Output: Representatives: $\mathbf{r}_1, \mathbf{r}_2, \ldots, \mathbf{r}_k$ with $k < N_{\text{rep}}$

it is separately applied on two or more parts of the dataset. The representatives from the parts are added together. Potentially, the process can be repeated with only the found representatives, hence a hierarchical divide-and-conquer strategy.

HSR needs two parameters: the maximum number of representatives N_{rep} and the branching factor h. Each recursion, SMRS is applied on each of the h parts. The representatives from each of the parts are combined and HSR is applied again if there are more representatives than N_{rep}. Using more parts reduces the computation load since SMRS is applied on a smaller dataset. Also, increasing the maximum number of representatives reduces the computational load because HSR keeps applying SMRS until there are less representatives then the maximum.

For the empirical tests however only one recursion was used. Thus SMRS was applied on the parts and then applied once more on the representatives. In this way the results were very similar to the ones obtained by SMRS. Consequently, the parameter N_{rep} was not used. Furthermore, for all experiments in this research the branching factor was $h = 2$.

2.1.2 Row-sparse subspace clustering

Row-sparse subspace clustering (RSSC) uses the results of SMRS or HSR to initialize the in-sample data for SSSC (see Algorithm 5). There are two immediate possibilities when using SMRS to initiate the in-sample set: the representatives or the non-representatives. The representatives are data points that are likely to be on the convex hull, and thus capture the whole variation of a subspace. Using these as a dictionary to represent the out-of-sample set guarantees that every data point can be represented with summed coefficients lower than one. In other words, the representatives are ideal because the SSSC out-of-sample regularized linear regression can easily expresses all data points.

Algorithm 5 Row-sparse subspace clustering

Input: A set of data points $\mathbf{Y} = \{\mathbf{y}_i\}_{i=1}^N$ from a union of n subspaces $\{\mathcal{S}_i\}_{i=1}^n$
Input: The ridge regression parameter λ
$\mathbf{r} = \text{SMRS}(\mathbf{Y})$ or $\mathbf{r} = \text{HSR}(\mathbf{Y})$
$\mathbf{X}^{\text{in/out}} = \{\mathbf{Y}_{\mathbf{r}_i} \mid \mathbf{r}_i \in \mathbf{r}\}$
$\mathbf{X}^{\text{out/in}} = \{\mathbf{Y}_{\mathbf{r}_i} \mid \mathbf{r}_i \notin \mathbf{r} \wedge \mathbf{r}_i \in \{1, 2, \ldots, N\}\}$
$\{\mathbf{X}_1^{\text{in}}, \mathbf{X}_2^{\text{in}}, \ldots \mathbf{X}_N^{\text{in}}\} = \text{SSC}(\mathbf{X}^{\text{in}})$
$\mathbf{C}_i^* = (\mathbf{X}^{\text{in}\top}\mathbf{X}^{\text{in}} + \lambda\mathbf{I}) \cdot \mathbf{X}^{\text{in}\top}\mathbf{X}^{\text{out}}$
Calculate the normalized residuals of $\bar{x}^i \in \bar{\mathbf{X}}$ over all classes by

$$r_j(\bar{x}_i) = \frac{||\bar{\mathbf{x}}_i - \mathbf{X}\delta_j(\mathbf{c}_i^*)||_2}{||\delta_j(\mathbf{c}_i^*)||_2}$$

$\text{identity}(\mathbf{X}_i^{\text{out}}) = \underset{j}{\text{argmin}}\, r_j(\mathbf{X}_i^{\text{out}})$.
Output: Segmentation of the data: $\mathbf{Y}_1$, $\mathbf{Y}_2$, ..., $\mathbf{Y}_n$

However, representatives are on the convex hull and thus closer to other subspaces than data points inside the convex hull. Furthermore, data points on the convex hull are hard to represent as other data points on the convex hull, i.e. need linear combinations that sum to more than one. This makes the SSSC in-sample clustering step harder because the subspace structure is harder to detect. A bad in-sample clustering does not predict anything good for the out-of-sample clustering. Summarizing, with representatives the in-sample clustering is likely to be hard but with the right in-sample cluster labels the out-of-sample step works well.

Using the non-representatives as in-sample set increases the odds that the in-sample set is clustered correctly. In-sample data points are closer together but more importantly they can more easily be represented as a linear combination of each other. Thus the in-sample clustering labels become more accurate. However, using non-representatives increases the chances that a dimension of a subspace is not modelled at all by the in-sample set and thus the out-of-sample step of SSSC becomes less accurate.

Consequently, it is unknown if either random, representative or non-representative data points work best for the in-sample set of SSSC. Therefore all three methods were used and compared to each other.

The RSSC algorithm needs several parameters. The α value is used to compute the (non-)representatives using SMRS. The ridge regression parameter λ and tolerance δ are used by SSSC. These parameter are estimated in the second experiment, that is described in Section 2.4.

Another option to use the representatives for high-dimensional data clustering

Name	Symbol
# Data points in each subspace	N_ℓ
# Subspaces	n
Dimension of subspaces ℓ	d_ℓ
Dimension of dataset	D
Smallest principal angle between subspaces $\mathcal{S}_i$ and $\mathcal{S}_j$	$\cos(\theta_{ij})$
Noise	ϵ^2

Table 2.1: Interesting parameters of linear subspace

is to cluster the row-sparse coefficient matrix right away. However, the row-sparse constraint does not guarantee the needed block-sparse structure because an ℓ_2 norm on the rows is used. The example matrix in Figure 1.6 has both row-sparsity and block-sparsity but was cherry picked because of it. Experiments that were done in the preparation of this research that used the SMRS coefficient matrix did not have a good performance.

2.2 Datasets

The runtime and accuracy of the algorithms was measured with three different high-dimensional datasets: generated linear subspaces, the Extended Yale B dataset and the Hopkins 155 dataset. These are described in the following sections.

2.2.1 Generated linear subspaces

The cosine similarity between the subspaces and the underlying data distribution for each subspace are important for SSC. Elhamifar and Vidal (2013) showed that when the smallest variation in the subspace is larger than the largest cosine similarity between subspaces times some constant, SSC correctly recovers the subspace structure (see Section 1.3.1). In the following a method is described to generated linear subspaces with a fixed underlying structure, i.e. a fixed cosine similarity between the subspaces and additional noise.

The parameters for generating linear subspaces are shown in Table 2.1. Most important were the cosine similarity between the subspaces $\cos(\theta_{ij})$ and the noise ϵ^2.

The data was generated in three steps: (a) For each subspace a low-dimensional representation were sampled from a normal distribution, (b) a random rotation with fixed angles between the subspaces was iteratively found and (c) noise was added to the high-dimensional data.

The low-dimensional representation $\mathbf{U}_\ell \in \mathbb{R}^{d \times N_\ell}$ was sampled from a uniform distribution between zero and one.

$$\mathbf{U}_\ell = \mathcal{U}(0, 1) \tag{2.1}$$

The rotation were iteratively created. The initial rotation $\mathbf{R}_1 \in \mathbb{R}_\ell^{D \times d_\ell}$ was randomly created using the first d_1 columns of an orthogonal-triangular decomposition[1] of a random $D \times D$ matrix. The next rotations were a weighted combination of the null space $\text{Null}(\mathbf{R}_{<\ell})$ and the average of all the current rotations normalized to unit length $\mathbf{A}'$:

$$\mathbf{R}_\ell = \sin(\gamma)\text{Null}(\mathbf{R}_{<\ell}) + \cos(\gamma)\mathbf{A}' \tag{2.2}$$

Since both the null space and the normalized average of all current rotations had unit length, the new rotation also had unit length because $\sin(\gamma)$ and $\cos(\gamma)$ are on the unit circle. γ was being computed from the unnormalized averages of all rotations $\mathbf{A}$, a arbitrary rotation $\mathbf{R}_1$ (in this case the first) and the cosine similarity between the subspaces $\cos(\theta_{ij})$:

$$\cos(\gamma) = \frac{-2 - 2\cos(\theta_{ij}) + \|\mathbf{A}\|_2^2 - \|\mathbf{R}_1 - \mathbf{A}\|_2^2}{2\|\mathbf{A}\|_2} \tag{2.3}$$

where $\mathbf{A}'$ is $\mathbf{A}$ with normalized columns, and $\|X\|_2$ is the euclidean length of X. Since the vectors from the null space have unit length and the norm

Finally, the low-dimensional distribution and rotation were combined and noise was added:

$$X_\ell = R_\ell U_\ell^\top + E \quad \text{with} \quad E \in \mathcal{N}(\mathbf{0}, \epsilon^2) \tag{2.4}$$

An example of a two dimensional subspace of a three dimensional dataspace is shown in Figure 2.1. The convex hull is also plotted. Since there was no noise and the subspace were two dimensional, the convex hull was also two dimensional and shows that the subspace is a rotated plane.

The default parameters for this dataset were $N = 500, D = 2000, \cos(\theta_{ij}) = 0.5, \epsilon^2 = 0.1$. Unless when running with different cosine similarities or noise values these parameters were used.

2.2.2 Extended Yale B

The Extended Yale Face Database B (Lee et al., 2005; Georghiades et al., 2001) dataset consists of 21888 images of 38 subjects with nine different lightning conditions, thus 64 images per subject. An example of pictures taken from subjects is in Figure 2.2. Each data points is a cropped face with in total 2016 pixels.

[1] See the Matlab function: `http://nl.mathworks.com/help/matlab/ref/qr.html`

Regarding to the noise and principal angles the Extended Yale B dataset has the following properties (Elhamifar and Vidal, 2013). The principal angles between any pair of subspaces are between 10° and 20° and thus well separated. The singular values rapidly decrease until the ninth one, after that the singular values are low but not zero, i.e. noise. It is thus likely that the faces from the Extended Yale B dataset live in a low-dimensional subspace with nine dimensions. When looking at the nearest neighbours for each point, the majority of the 7th or more nearest neighbour is in another subspace. This is a good indication that a distance measure is not suited to cluster this dataset.

Experiments with this dataset were repeated 100 times. To get 100 repeats this research only used a subset of subjects at the time. The n subjects are randomly selected from 38 subjects. In total $\binom{38}{n}$ were possible. In the case of using five subjects at the time this results in 501942 possible combinations but these datasets were not independent.

2.2.3 Hopkins 155

The Hopkins 155 dataset (Tron and Vidal, 2007) consists of 155 video frames with image feature trajectories (see Figure 2.3). These trajectories are most reliable for the chessboard videos, wherein chessboards rotate or translate or both. Other videos show moving cars, people or arms. In part of the movies the camera also moves in a unstable way.

The trajectories are divided into clusters by hand. The trajectories on a single object are assigned to a cluster. Each video frame has either two or three different clusters. Some of these clusters belong to moving objects, others belong to fixed objects but since the camera also moves these are hard to separate.

The noise and principal angle properties are as follows (Elhamifar and Vidal, 2013). The principal angles between any pair of subspaces is under 5°, i.e. the subspaces are very similar. The singular values rapidly decrease until the 4th one, after that they are almost zero meaning that there is little noise. It is thus likely that the trajectories have a 4 dimensional underlying structure. The subspaces are well separated because less than 20% of the nearest neighbours, up to the 30th, are in another subspace.

Experiments with this dataset were repeated 155 times. Each of the datasets in the Hopkins 155 set was used one time. These datasets were thus independently.

2.3 Measures

The representatives from SMRS and HSR were compared using three different statistics described in Section 2.3.1. The clustering performance of the SSC, SSSC

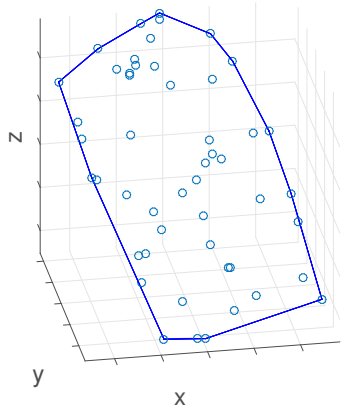

Figure 2.1: Generated two-dimensional subspace in a three-dimensional dataspace

Figure 2.2: 28 subjects from the Extended Yale B dataset

Figure 2.3: Example image from the Hopkins 155 dataset enhanced with features

and HSR were compared using the normalized mutual information (Cai et al., 2005) and the subspace clustering error (Elhamifar and Vidal, 2013). These are described in Section 2.3.2. To obtain matches between the ground-truth labels and the computed labels the cluster assignments were matched, which is described in Section 2.3.3.

2.3.1 Representatives accuracy, precision and correlation

To compare two sets of representatives two categories of measurements were used. The first were the accuracy and precision and those measured how many representatives that were found by SMRS were also found by HSR. Representatives that were found by both algorithms were true positives, representatives that were only found by SMRS were false negatives, representatives that were found only by HSR were false positives and representatives not found by both were true negatives. The accuracy and precision were calculated as:

$$\text{accuracy} = \frac{\text{true positives} + \text{true negatives}}{N} \tag{2.5}$$

$$\text{precision} = \frac{\text{true positives}}{\text{true positives} + \text{false positives}} \tag{2.6}$$

A second category measured the similarity in the ordering of the representatives. Remember that the representatives were ordered based on their importance. This measure is only computed for representatives that were truly positive because both algorithms should report the representatives. To compare the ordering the normalized index was computed by dividing the index i_j by the total number of data points N (including non-representatives):

$$\iota = \frac{i_j}{N} \tag{2.7}$$

The correlation between the two sets of normalized indexes was computed as a measure of the similarity between the orderings.

2.3.2 Cluster error and normalized mutual information

The goodness of the clustering was measured with the subspace clustering error and the normalized mutual information. These measures were earlier used by Elhamifar and Vidal (2013) and Peng et al. (2013b) to assess their clustering results.

The subspace clustering error is simply the proportion of misclassified points:

$$\text{subspace clustering error} = \frac{\#\text{ misclassified points}}{N} \tag{2.8}$$

The normalized mutual information MI (Cai et al., 2005) between the found clustering S' and ground truth clustering S also incorporates knowledge about the amount and size of the clusters. It is computed as:

$$\mathrm{MI}^*(S, S') = \sum_{s_i \in S, s'_i \in S'} p(s, s') \log_2 \frac{p(s, s')}{p(s) \cdot p(s')} \tag{2.9}$$

$$\mathrm{MI}(S, S') = \frac{\mathrm{MI}^*(S, S')}{\max(H(S), H(S'))} \tag{2.10}$$

where $p(s, s')$ is the probability that an arbitrarily selected data point belongs to clusters s and s', $p(s)$ the probability that an arbitrarily selected data point belongs to s and alike for s'. $H(S)$ and $H(S')$ are the entropy's of S and S', respectively. The value of MI ranges from 0 to 1 and those values mean respectively that the clusters are totally different of the identical.

2.3.3 Matching cluster assignments

To obtain the normalized mutual information and subspace clustering error a matching between the found clusters and ground truth clusters needed to be found.

The Hungarian method (Kuhn, 1955) finds a minimal weighted matching of a bipartite graph. The vertices's are the found clusters and the ground truth clustering. The weights are defined as:

$$w(S, S') = \sum_{c_i \in S} \delta(c_i \notin S') + \sum_{c_j \in S'} \delta(c_j \notin S) \tag{2.11}$$

where $\delta(x)$ is the identity function that equals one if x is true and zero if x is false. Thus, the weight is the number of data points that is in the found cluster but not in the ground truth cluster and vice versa.

2.4 Experiments

Three experiments were done to compare the different algorithms. The first experiment compared RSSC with HSR. The second experiment was used to compute the optimal parameters for each algorithm. The third experiment compared the clustering performance of SSC, SSSC and RSSC. These experiments are now described in more detail.

Dataset	α's
Generated linear subspaces ($\epsilon = 0.0$)	$\{2, 3, \ldots, 20\}$
Generated linear subspaces ($\epsilon = 0.1$)	$\{1.05, 1.10, \ldots, 1.40\}$
Extended Yale B	$\{5, 10, \ldots, 50, 100, \ldots, 500\}$
Hopkins 155	$\{50, 100, \ldots, 500, 600, \ldots, 1500\}$

Table 2.2: The α values for each dataset used to compare the representatives of SMRS to the representatives of HSR

2.4.1 Experiment 1: representatives search

The aim of this experiment was to compare the results of SMRS and HSR and see if they were similar. There were two perspectives on this similarity. First there was the similarity in the found representatives. This was tested using the accuracy and precision of the representatives of HSR, using the representatives from SMRS as ground truth. Secondly, there was the similarity between the importance of the representatives. This was tested using the correlation between the normalized indexes of the representatives from SMRS and HSR.

The results were obtained for the Extended Yale B dataset, the Hopkins 155 dataset and the generated linear subspaces without noise and much noise ($\epsilon = 0.1$). Because it was expected that the amount of representatives influenced the accuracy and correlation between the two sets of representatives multiple α values were used for each dataset. The used α values were reported in Table 2.2. For each of the α-dataset combinations 100 experiments were done and the results were stacked.

Ideally the normalized indexes of the two algorithms were highly correlated but also had a zero-mean distributed error. To test for the zero-mean distributed error, an one-sample t-test of the difference between the normalized indexes was performed.

2.4.2 Experiment 2: parameter optimization

For a fair comparison between the algorithms, each one should be using the optimal parameters for a particular dataset. This experiment found these optimal parameters for SSC, SSSC, RSSC$^{\text{rep}}$ and RSSC$^{\text{no}}$ for the generated linear subspaces, the Extended Yale B dataset and the Hopkins 155 dataset.

Table 2.3 shows the different parameters settings that were used for each algorithm for each dataset. The α values for both RSSC algorithms were chosen such that the number of found (non-)representatives ranges from very few to almost all. Also note that the RSSC algorithm will use SMRS to find the representatives since the aim is to validate that representatives can be used to get a fast and reliable clustering. Also note that λ and δ for RSSC was not optimized over. First the op-

timal parameters for SSSC were computed and then these optimal values for SSSC for λ and δ were used for RSSC. Lastly, note that there was no distinction between RSSC with representatives and non-representatives because both variations were optimized over with the same parameters.

The experiments with generated linear subspaces were repeated 100 times. The generated linear subspaces had 501 data points, a 10 dimensional lower manifold for each subspace that was embedded in 2000 dimensions. Furthermore, the cosine similarity between the subspaces was 0.5 and the standard deviation of the Gaussian noise was 0.1. Experiments with the Extended Yale B dataset were also repeated 100 times. Subsets of 5 subjects from the 38 subjects of the Extended Yale B dataset were used. The 155 datasets of the Hopkins 155 dataset were all used once.

The results were aggregated for each algorithm for each set of parameters for each dataset. A Kolmogorov-Smirnov test was used to test if the distributions are normal. Since the error measurements were between 0.0 and 1.0 and many runs were expected to be errorless, the distributions were expected to be non-normal. In this case, the non-parametric Kruskal-Wallis test was used to test if the parameter settings differ significantly. In the unlikely case of a normal distribution, an one-way ANOVA was used. Respectively, a Wilcoxon test or Turkey's range test was used for post hoc analysis if the group means differ significantly.

2.4.3 Experiment 3: clustering performance

Using the optimal parameters from Experiment 2 (see the results in Section 4.3) three algorithms for high dimensional data clustering were compared. These algorithms were SSSC, RSSC$^{\text{rep}}$ and RSSC$^{\text{no}}$. The used parameters are shown in Table 2.4.

Note that for each of the in-sample algorithms only 10% or 20% (in the case of the Hopkins 155 dataset) of the data points were used as in-sample. For the generated linear subspaces this $501 \times 0.1 \approx 50$ was more than the required $(d+1) \times n = (10+1) \times 3 = 33$ data points to represent each subspace sufficiently. For the Extended Yale B this $192 \times 0.1 \approx 19$ was less than the required $(9+1) \times 5 = 50$ data points to describe each subspace. For the Hopkins 155 dataset most of the times these data points were sufficient to describe all the subspaces, but the Hopkins 155 dataset has different sizes.

Like the experiments for parameter optimization the generated linear subspaces had 500 data points, a 10 dimensional lower manifold for each subspace that was embedded in 2000 dimensions. Furthermore, the cosine similarity between the subspaces was 0.5 and the standard deviation of the Gaussian noise was 0.1. For the Extend Yale B dataset again random sampling of 5 subsets out of 38 was used. Experiments with these two datasets were repeated 100 times. The 155 datasets

	Generated linear subspaces	Extended Yale B	Hopkins 155
SSC	$\alpha = \{2, 3 \ldots, 10, 20, \ldots, 100, 110, \ldots, 1000\}$ affine $= 0$ outliers $= 0$ $\rho = 1.0$	$\alpha = 20$ affine $= 0$ outliers $= 1$ $\rho = 1.0$	$\alpha = 800$ affine $= 1$ outliers $= 0$ $\rho = 0.7$
SSSC	$\lambda = \{10^{-4}, 10^{-5}, 10^{-6}, 10^{-7}, 10^{-8}\}$ $\delta = \{0, 0.1, 0.01, 0.001, 0.0001\}$ $p = 10\%$	$\lambda = 10^{-7}$ $\delta = 10^{-3}$ $p = 10\%$	$\lambda = \{10^{-4}, 10^{-5}, 10^{-6}, 10^{-7}, 10^{-8}\}$ $\delta = \{0, 0.1, 0.01, 0.001, 0.0001\}$ $p = 20\%$
RSSC	$\alpha = \{1.01, 1.05, 1.1, \ldots, 1.5, 1.6, \ldots, 2.0\}$ affine $= 0$ $\lambda = 10^{-4}$ $\delta = 10^{-3}$	$\alpha = \{5, 6, \ldots, 10, 15,$ $\ldots, 50, 60, \ldots, 300\}$ affine $= 0$ $\lambda = 10^{-7}$ $\delta = 10^{-3}$	$\alpha = \{20, 30, \ldots, 100, 200, \ldots, 1000\}$ affine $= 1$ $\lambda = 10^{-4}$ $\delta = 10^{-4}$

Table 2.3: Parameters ranges for which the best parameter combinations are found during experiment 1

	Generated linear subspaces	Extended Yale B	Hopkins 155
SSC	$\alpha = 20$ affine $= 0$ outliers $= 0$ $\rho = 1.0$	$\alpha = 20$ affine $= 0$ outliers $= 1$ $\rho = 1.0$	$\alpha = 800$ affine $= 1$ outliers $= 0$ $\rho = 0.7$
SSSC	$\lambda = 10^{-4}$ $\delta = 0.001$ $p = 10\%$	$\lambda = 10^{-7}$ $\delta = 10^{-3}$ $p = 10\%$	$\lambda = 10^{-4}$ $\delta = 10^{-4}$ $p = 20\%$
RSSC$^{\text{rep}}$	$\alpha = 1.05$ affine $= 0$ $\lambda = 10^{-4}$ $\delta = 10^{-3}$ $p = 10\%$	$\alpha = 5$ affine $= 0$ $\lambda = 10^{-7}$ $\delta = 10^{-3}$ $p = 10\%$	$\alpha = 800$ affine $= 1$ $\lambda = 10^{-4}$ $\delta = 10^{-4}$ $p = 20\%$
RSSC$^{\text{no}}$	$\alpha = 1.80$ affine $= 0$ $\lambda = 10^{-4}$ $\delta = 10^{-3}$ $p = 10\%$	$\alpha = 120$ affine $= 0$ $\lambda = 10^{-7}$ $\delta = 10^{-3}$ $p = 10\%$	$\alpha = 50$ affine $= 1$ $\lambda = 10^{-4}$ $\delta = 10^{-4}$ $p = 10\%$

Table 2.4: Parameters that were used during experiment 3

of the Hopkins 155 dataset were all used once.

The clustering error and mutual info were tested for normality using a Kolmogorov-Smirnov test. In the normal case the distributions were tested for differences using a ANOVA, in the non-normal case an Kruskal-Wallis test was used. Pairwise testing was, respectively, done with a Turkey's range or Wilcoxon test.

One additional experiment was done to interpret the results better. RSSC$^{\text{rep}}$ and SSSC were applied to the generated linear subspaces with different cosine similarities ranging from zero to one. Furthermore, the size of the in-sample set was varied between $\{0.5, 1.0, 1.5\} \times (d+1) \times n$. With only $0.5 \times (d+1) \times n$ there were not enough data points to reconstruct the data points perfectly and it was expected that it was more important to select the right data points. Consequently, the RSSC$^{\text{rep}}$ algorithm should have had an advantageous above SSSC.

Chapter 3

Theoretical results

The results of SMRS and HSR can be compared using the notion of convex hull because Elhamifar and Vidal (2013) showed that SMRS is an approximation of the convex hull. Using these convex hull properties the equivalence of SMRS and HSR is derived in Section 3.1. Thereafter, in the second section of this chapter the complexity of SMRS and HSR are compared to see if HSR is preferable.

3.1 Equivalence of sparse modeling representatives selection and hierarchical sparse representatives

Elhamifar and Vidal (2013) showed that the representatives from the SMRS algorithm are on the convex hull of the dataset. The concept and properties of these convex hulls are earlier described in Section 1.1.2. In the following sections these properties are used to reason about the convex hull of merged sets and its implications for the equivalence of SMRS and HSR.

3.1.1 Convex hulls

Summarizing Section 1.1.2, the convex hull Area(Conv($\mathbf{Y}$)) of a set of data points $\mathbf{Y}$ is the region that is spanned by a linear weighting of the data points in $\mathbf{Y}$. Intuitively it can be thought of as stretching a rubber band around all the data points and letting it go. The rubber band contracts and hooks on the data points that are on the convex hull (see Figure 1.2). The set of data points that are on the convex hull is denoted with Conv($\mathbf{Y}$). The area spanned by these datapoints (or all datapoints) is denoted as Area(Conv($\mathbf{Y}$)).

Adding a data point inside the convex hull does not change the convex hull. Also, adding a data point that is on the convex hull does not change the convex hull because these data points can be represented as a linear weighting of the other data points. Only data points outside the convex hull that are added to the dataset change the convex hull. The new data point becomes part of the convex hull and some other data points that were on the convex hull might now be inside the convex hull. This is because they might be represented as a linear combination of the new data point and other data points that already were on the convex hull (with $\sum \lambda_i = 1$). In short, adding data points to a dataset can increase the area but never decrease the area spanned by the convex hull. More formally:

$$\text{Area}(\text{Conv}(A \cup B)) \geq \text{Area}(\text{Conv}(A)) \tag{3.1}$$

where the $\geq$ means that one area dominates the other, i.e. the second area is encapsulated by the first.

Each data point inside the convex hull of one of the two subsets can be linearly represented by data points on the convex hull of the subset. These subset convex hull data points are also in the merged set and therefore data points that are inside the convex hull of a subset are also inside the convex hull of a merged set. Thus only data points that are on the convex hull of one of the subsets can be on the convex hull of the merged set. More formally:

$$\text{Conv}(\mathcal{S}_1 \cup \mathcal{S}_2) \subseteq \text{Conv}(\mathcal{S}_1) \cup \text{Conv}(\mathcal{S}_2) \tag{3.2}$$

It is also true that the area spanned by the convex hull of two merged sets is larger than the merged area of the convex hulls of those sets. First we can conclude that if B is a subset of A, than the convex hull of A is equal or larger than B because of Equation 3.1:

$$A \supseteq B \rightarrow \text{Area}(\text{Conv}(A)) \geq \text{Area}(\text{Conv}(B)) \tag{3.3}$$

A merged set is the union of two subsets, and thus we know that

$$\mathcal{S}_1 \cup \mathcal{S}_2 \supseteq \mathcal{S}_1 \tag{3.4}$$

and we can conclude that

$$\text{Area}(\text{Conv}(\mathcal{S}_1 \cup \mathcal{S}_2)) \geq \text{Area}(\text{Conv}(\mathcal{S}_1)) \tag{3.5}$$

The same can be done for $\mathcal{S}_2$ and thus it is true that

$$\text{Area}(\text{Conv}(\mathcal{S}_1 \cup \mathcal{S}_2)) \geq \text{Area}(\text{Conv}(\mathcal{S}_1)) \cup \text{Area}(\text{Conv}(\mathcal{S}_2)) \tag{3.6}$$

where $\text{Area}(A) \cup \text{Area}(B)$ means the union of region $\text{Area}(A)$ and $\text{Area}(B)$.

3.1.2 Implications for hierarchical sparse representatives

HSR applies SMRS on parts of the dataset. Elhamifar and Vidal (2013) showed that the representatives are an approximation of the convex hull. Using Equation 3.6 and Equation 3.2 it is certain that the representatives that are found by applying SMRS on the whole datasets are also found when applying SMRS on parts of the dataset.

3.2 Complexity analysis

In the next two sections (Section 3.2.1 and Section 3.2.2) the complexity of both SMRS and HSR are derived. Then, in Section 3.2.3 the conditions under which HSR is faster than SMRS are computed.

3.2.1 Sparse modeling representatives selection

Assuming that the complexity of optimization of SMRS is similar to the optimization of SSC the known complexity of the homotopy optimizer can be used. The homotopy optimizer (Osborne et al., 2000) is currently the fastest known method for solving the lasso (Yang et al., 2010). It has complexity (Peng et al., 2013a):

$$\mathcal{O}(tn^2m^2 + tn^3m) \tag{3.7}$$

but when assuming that $m < n$ the complexity reduces to:

$$\mathcal{O}(tn^3m) \tag{3.8}$$

Also, the complexity of the eigenvector computation ($\mathcal{O}(n^3)$) can be ignored. Furthermore, the complexity of t_k iterations of the k-means algorithm ($\mathcal{O}(nkmt_k)$) can be ignored knowing that $k < n$ is always true, since the number of clusters cannot be larger than the number of data points, and assuming that $t_k \sim t$.

3.2.2 Hierarchical sparse representatives

Instead of using sets of data points of size n, HSR uses sets of size $s < n$. Therefore it needs to apply SMRS n/s times to get row-sparse data points for the whole set. However, depending on the portion of the number of row-sparse data points ρ, the process needs to be applied again on the union of the row-sparse data points from different sets until a satisfactory number of data points remain. Knowing that ρ is between 0.0 and 1.0, it can be concluded that the total parametrized complexity

of HSR is:

$$\begin{aligned}\mathcal{O}(\frac{n}{s}ts^3m + \rho\frac{n}{s}ts^3m + \rho^2\frac{n}{s}ts^3m + \dots) &= \mathcal{O}(\frac{n}{s}ts^3m(\rho^0 + \rho^1 + \rho^2 + \dots)) \\ &= \mathcal{O}(\frac{n}{s}ts^3m\frac{1}{1-\rho}) \\ &= \mathcal{O}(\frac{nts^2m}{1-\rho})\end{aligned} \tag{3.9}$$

3.2.3 Comparing sparse modeling representatives selection and hierarchical sparse representatives

Given the portion of row-sparse data points the size of the subsets can be computed for which SMRS and HSR have equal complexity. Thus, setting the complexities equal results in:

$$\begin{aligned}\mathcal{O}(tn^3m) &> \mathcal{O}(\frac{nts^2m}{1-\rho}) \\ \mathcal{O}(n^2) &> \mathcal{O}(\frac{s^2}{1-\rho}) \\ \mathcal{O}(n^2(1-\rho)) &> \mathcal{O}(s^2) \\ \mathcal{O}(\sqrt{n^2(1-\rho)}) &> \mathcal{O}(s) \\ \mathcal{O}(n\sqrt{1-\rho}) &> \mathcal{O}(s)\end{aligned} \tag{3.10}$$

And thus, in order to get an improved complexity, the size of the subsets s in HSR should be:

$$s < n\sqrt{1-\rho} \tag{3.11}$$

This means that for $\rho < 0.75$ the size of the subsets s can be half the size of n and thus effectively splitting the original sized dataset repeatedly into two. Consequently, when splitting into more parts or when $\rho < 0.75$ the speed improvement is significant.

Chapter 4

Numerical results

4.1 Manipulation checks

To verify that the cosine similarity of the actual generated linear subspaces corresponded to the input parameter cosine similarity $\cos(\theta_{ij})$ the differences between the input and output cosine similarity were computed (See Figure 4.1). Note that the difference were very small, i.e. $\leq 10^{-2}$. Also note that the computed cosine similarity was never larger than the input cosine similarity.

The effect of changing the standard deviation of the noise σ on the measured noise is shown in Figure 4.2. The noise was measured over every subset $\mathbf{Y}_i$ as the sum of the singular values for indexes larger than d_i divided by the sum of all singular values. Note that the amount of noise increased as expected when ϵ^2's increased and that only little noise is needed (e.g. $\epsilon^2 = 0.1$) because the data is generated in $[0, 1]$.

4.2 Experiment 1: representatives search

The accuracy, precision and correlation of the representatives are shown from Figure 4.3 to Figure 4.6. For all datasets the accuracy, precision and correlation increased when more representatives were found.

For the generated linear subspaces without noise (Figure 4.3), only few representatives ($< 23\%$) were found. Increasing the α further did not increase the number of representatives. The accuracy was high (≥ 0.95) but mainly influence by the large part of non-representatives. The precision increased slightly from 0.80 (2.8% representatives) to 0.89 (22.9% representatives). The correlation increased from 0.56 to 0.81. All error distributions had a significant non-zero mean ($p < 0.05$) except for $\alpha = 2$ and $\alpha = 5$.

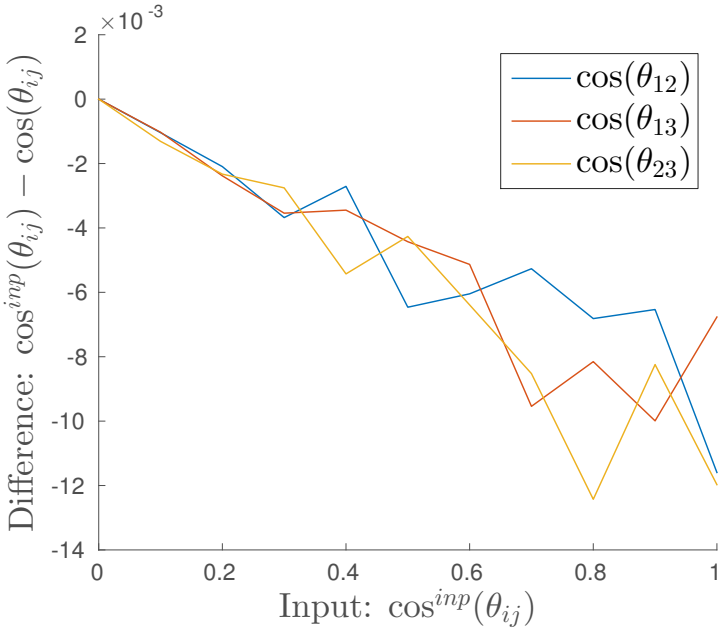

Figure 4.1: Difference between the input cosine similarity $\cos(\theta_{ij})$ and the computed cosine similarity between three subspaces. This data is generated with $N = 100, D = 500, d = 10, n = 3, \epsilon = 0.0$.

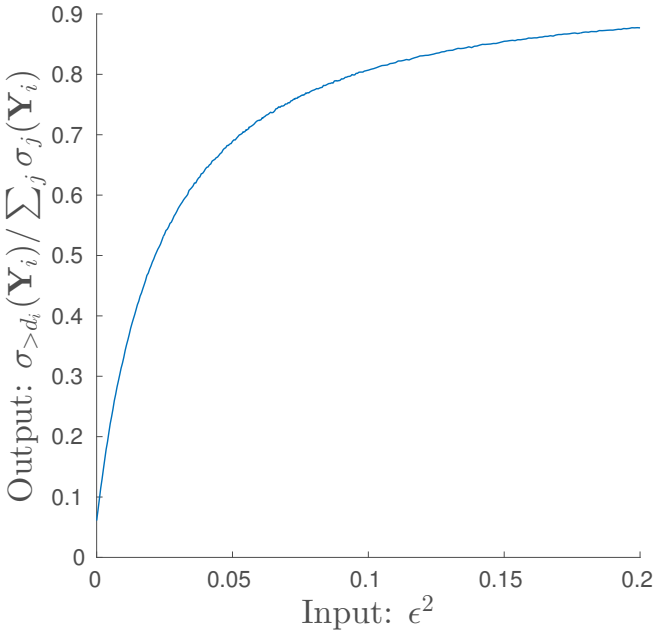

Figure 4.2: Part of data explained by noise of each subset $\tilde{\mathbf{Y}}_{ii}$ of a subspace $\mathcal{S}_i$ for different input noises ϵ^2. The results were generated with $N = 40, D = 200, d = 3, n = 3$.

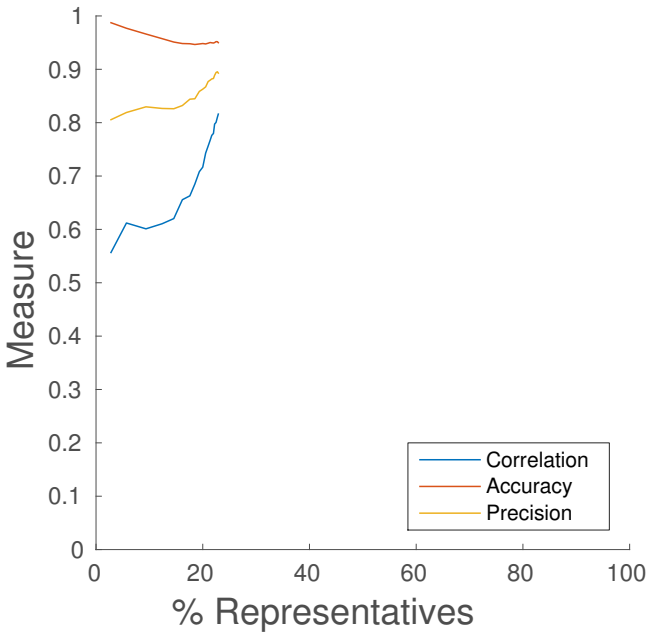

Figure 4.3: The accuracy, precision and correlation of the HSR representatives compared to the SMRS representatives for different values of α on generated linear subspaces without noise

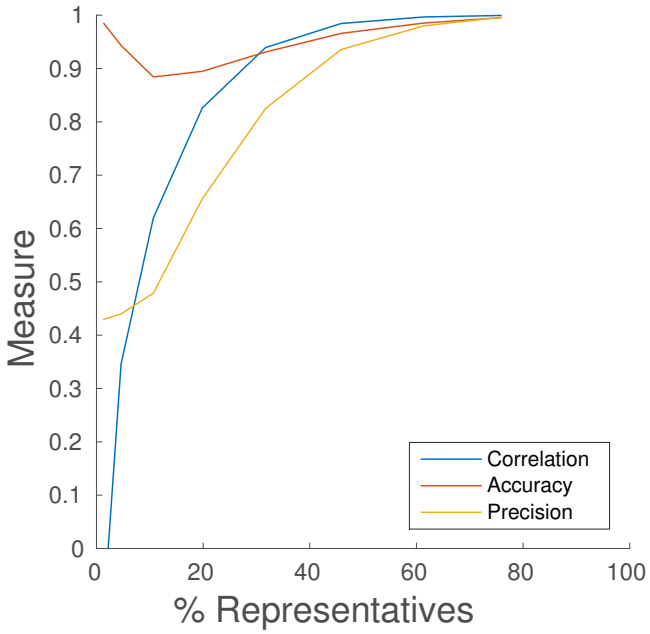

Figure 4.4: The accuracy, precision and correlation of the HSR representatives compared to the SMRS representatives for different values of α on generated linear subspaces with noise $\epsilon^2 = 0.1$

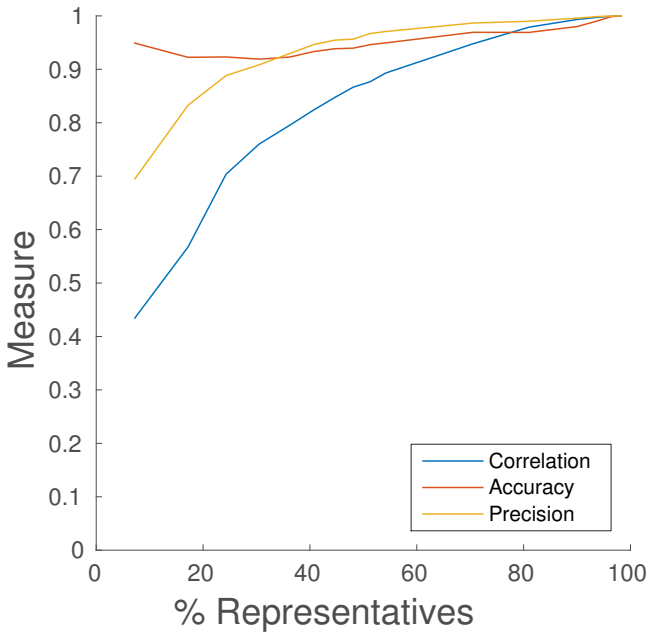

Figure 4.5: The accuracy, precision and correlation of the HSR representatives compared to the SMRS representatives for different values of α on the Extended Yale B dataset

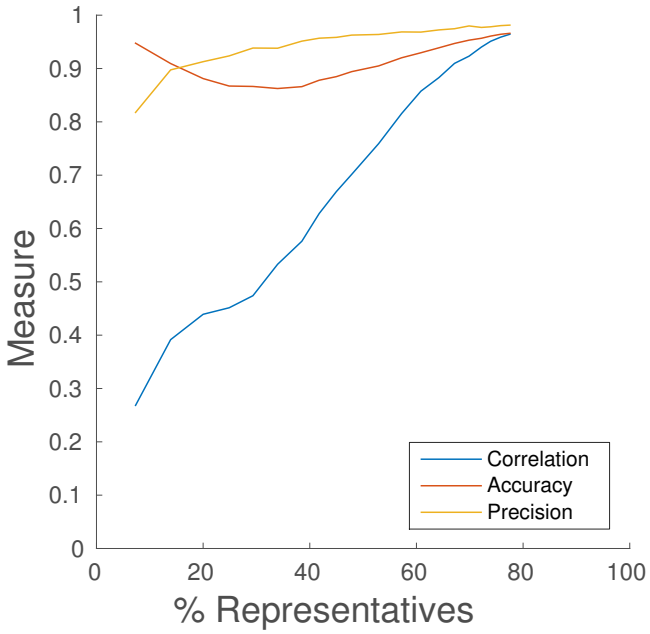

Figure 4.6: The accuracy, precision and correlation of the HSR representatives compared to the SMRS representatives for different values of α on the Hopkins 155 dataset

For the generated linear subspaces with noise (Figure 4.4) the number of representatives increased. The accuracy was high (≥ 0.94) in general but had a dip for α values 1.15 and 1.20 with 0.88 and 0.89 respectively. The precision increased steadily from 0.42 (1.3% representatives) to 0.99 (74.8% representatives). The correlation increased gative -0.11 (1.38% representatives) to 0.99 (74.8% representatives). All the error distributions had a significant non-zero mean ($p < 0.000$) except for $\alpha = 1.05$.

For the Extended Yale B dataset the number of representatives increased steadily (Figure 4.5). The accuracy was high (> 0.9). The precision increased when alpha increases from 0.69 (7.2% representatives) to 0.99 (98.2% representatives). The correlation increased from 0.43 (7.2% representatives) to 1.0 (98.2% representatives). All the error distributions had a significant non-zero mean ($p < 0.05$) except for $\alpha = 500$.

For the Hopkins 155 the performance measures were high or increased steadily when the number of representatives increased (Figure 4.6). The accuracy was high (> 0.9) for few and many representatives. But for intermediate amounts of representatives ($20\% < x < 50\%$) the accuracy was somewhat lower (> 0.86). The precision was high (> 0.9) for almost all amounts of representatives ($> 20\%$). The correlation increased from 0.26 (7.3% representatives) to 0.96 (77.5% representatives). All the error distributions had a significant non-zero mean ($p < 0.000$).

4.3 Experiment 2: parameter optimization

In this section the results of the parameter optimization are discussed. The corresponding figures are all in Appendix B for clarity.

4.3.1 Generated linear subspaces

The subspace clustering error for SSC with different values of α can be seen in Figure B.1. The Kolmogorov-Smirnov test indicated that both the subspace clustering error and normalized mutual info were not normally distribute. Using the Kruskal-Wallis test significant differences between the clustering error and mutual information of different α's were found ($p < 0.000$). The best α was chosen based on the lowest mean subspace clustering error and highest mean normalized mutual information. The best α for both the subspace clustering error and normalized mutual information was $\alpha = 20$. Other values for α that did not have significant different clustering error or mutual information than $\alpha = 20$ were $20 \leq \alpha \leq 50$. For further computations $\alpha = 20$ was chosen.

The results for optimizing the λ and tolerance parameter for SSSC are shown in Figure B.2. The group distributions of the clustering error and mutual information

differed significantly ($p < 0.000$) in a non-normal way ($p < 0.000$). The best parameter setting for both measures was $\lambda = 1e^{-4}$ and tolerance $= 1e^{-3}$. Other settings that did not significantly differed were tolerance $= \{1e^{-3}, 1e^{-4}\}$ for all λ's (both $p > 0.9$). For further computations $\lambda = 1e^{-4}$ and tolerance $= 1e^{-3}$ was used.

The results for optimizing α for RSSC$^{\text{rep}}$ are displayed in Figure B.3. The group distributions for clustering accuracy and mutual information were not significantly different (both $p > 0.3$) and both distributions are non-normal (both $p < 0.000$). For further computations $\alpha = 1.05$ was chosen.

The results for optimizing α for RSSC$^{\text{no}}$ are displayed in Figure B.4. The group distributions did significantly differed for the clustering error and mutual information (both $p < 0.000$) in a non-normal way (both $p < 0.000$). The best setting for both measures was $\alpha = 1.8$. Other settings that did not have a significant different distribution on one of the measures were $\alpha \leq 1.30$ and $1.5 \leq \alpha \leq 1.9$. For further computations $\alpha = 1.8$ was chosen.

4.3.2 Extended Yale B dataset

The results for optimizing α for RSSC$^{\text{rep}}$ for the Extended Yale B dataset are shown in Figure B.5. There was a significant difference in clustering error and mutual information distributions between different values of α (both $p < 0.000$). Both measures were non-normally distributed (both $p < 0.000$). The best setting was $\alpha = 5$ for both the clustering error and mutual information. Other settings that were not significantly different on both measures were $\alpha \leq 9$, $\alpha = 170$, $\alpha = 180$, $\alpha = 200$, $\alpha \geq 220$. For further computations $\alpha = 5$ was chosen.

The results for optimizing α for RSSC$^{\text{no}}$ are displayed in Figure B.6. There was a significant difference in clustering error and mutual information between different values of α (both $p < 0.000$). Both measures erenon-normally distributed (both $p < 0.000$). The best setting for the clustering error was $\alpha = 120$. For the mutual information $\alpha = 290$ had the best results. But these values for α were not significantly different on both measures ($p > 0.9$). Other values for alpha that were not significantly different on both measures were $30 \leq \alpha \leq 190$, $\alpha \geq 210$. For further computations $\alpha = 120$ was chosen.

4.3.3 Hopkins 155 dataset

Figure B.7 shows the result of the SSSC algorithm with different values for the tollerance and λ. There was a significant difference ($p < 0.000$) between their non-normal ($p < 0.000$) distributions. The best was tollerance $= 10e^{-4}$ and $\lambda = 10e^{-4}$. This setting was significantly different from all with tollerance $\leq 1e^{-2}$,

from {tollerance $= 1e^{-3}, \lambda = 1e^{-7}$} and from {tollerance $= 1e^{-4}, \lambda \leq 1e^{-7}$} For further computations {tollerance $= 1e^{-4}, \lambda = 1e^{-4}$} was used.

Figure B.8 shows the results of RSSC$^{\text{rep}}$ with different values for α. There was a significant difference ($p < 0.000$) between their non-normal ($p < 0.000$) distributions. The best setting was $\alpha = 800$. This setting had a significant different clustering error and mutual information from $50 \leq \alpha \leq 100$. For further computations $\alpha = 800$ was chosen.

Figure B.9 shows the results of RSSC$^{\text{no}}$ with different values for α. There was a significant difference ($p < 0.000$) between their non-normal ($p < 0.000$) distributions. The best setting for the clustering error was $\alpha = 50$, for the mutual information $\alpha = 70$. This setting had a significant different clustering error and mutual information from $300 \leq \alpha \leq 700$ and $\alpha = 1000$. For further computations $\alpha = 50$ was chosen.

4.4 Experiment 3: clustering performance

4.4.1 Generated linear subspaces

Figure 4.7 shows the distribution of errors for SSSC, RSSC$^{\text{rep}}$ and RSSC$^{\text{no}}$ for the generated linear subspaces. Note that the errors were very close to zero. SSSC had the best error ($\mu = 0.015, Q_2 = 0.010$) and mutual information ($\mu = 0.942, Q_2 = 0.952$).

Performance All performance measurement distributions were non-normal ($p < 0.000$) and therefore the Kruskal-Wallis test was used to test for significant differences. There were no significant differences between the error distributions ($p > 0.9$) and mutual info distributions ($p > 0.9$). The results for all three algorithms were significantly worse (all $p < 0.000$) than SSC for the clustering error ($\mu = 0.000, Q_2 = 0.000$) and mutual info ($\mu = 0.999, Q_2 = 1.000$).

Duration The duration of RSSC with SMRS or HSR is shown in Figure 4.8. These differences were significant ($p < 0.000$).

Cosine similarity vs. in-sample size The effect of the number of representatives combined with the cosine similarity is shown in Figure 4.9 and Figure 4.10. The clustering error was zero for all generated datasets with a cosine similarity lower than one and an in-sample size $\geq (d+1) \times n$. There were no significant differences between SSSC and RSSC$^{\text{rep}}$ except for the cells where $\cos(\theta_{ij}) = 0.8$ and the in-sample size is $0.5 \times (d+1) \times n$ and where $\cos(\theta_{ij}) = 0.9$ and the in-sample size is $(d+1) \times n$ with both $p < 0.05$.

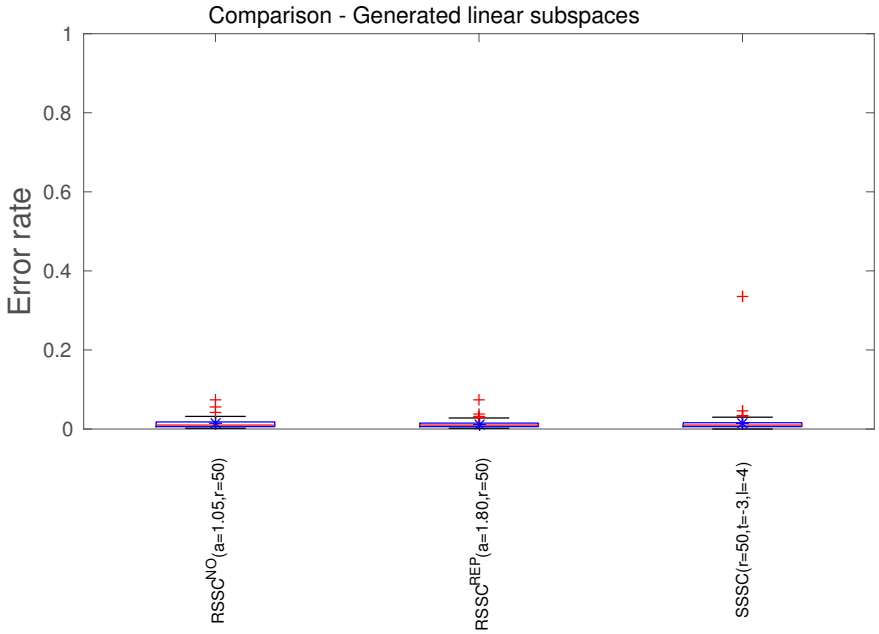

Figure 4.7: Distribution of errors on generated linear subspaces.

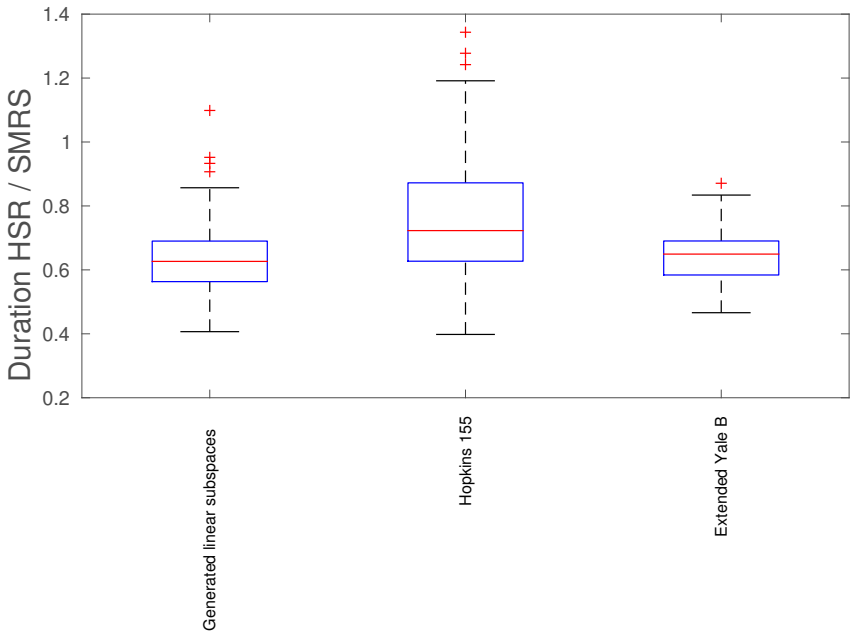

Figure 4.8: Difference in duration between the RSSC algorithm with SMRS and the RSSC algorithm with HSR for all datasets.

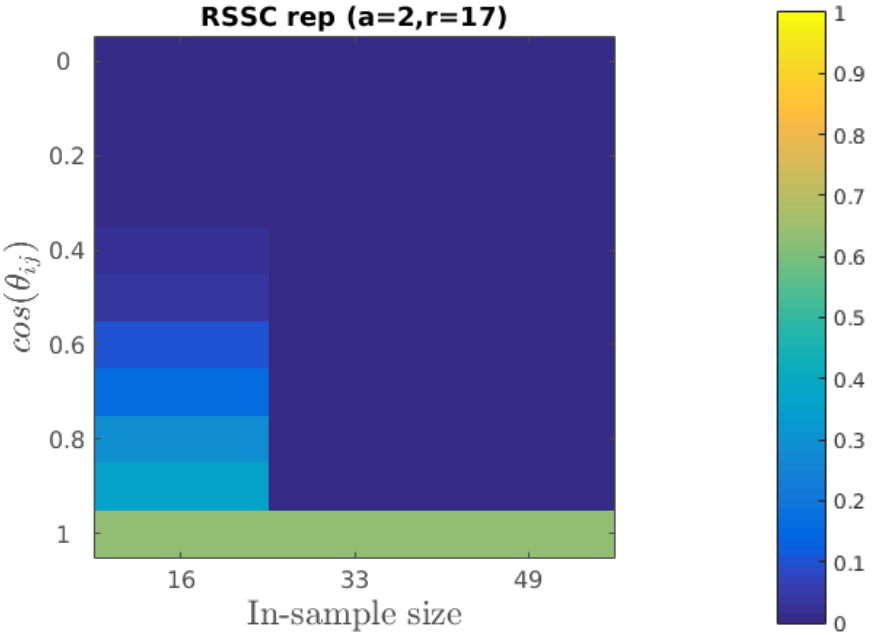

Figure 4.9: Clustering error of RSSC$^{\text{rep}}$ for generated linear subspaces with different cosine similarities, no noise, 501 data points and three ten-dimensional subspaces.

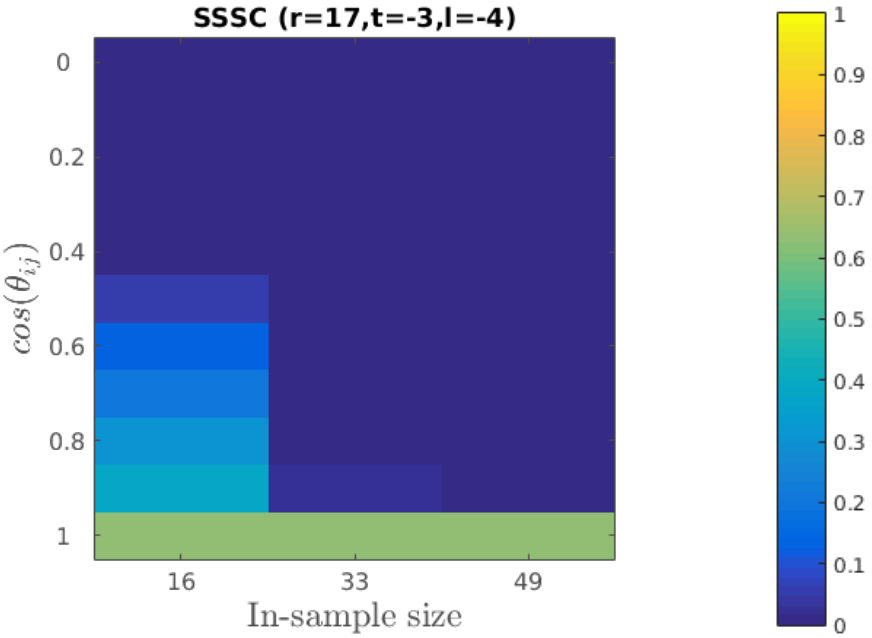

Figure 4.10: Clustering error of SSSC for generated linear subspaces with different cosine similarities, no noise, 501 data points and three ten-dimensional subspaces.

4.4.2 Extended Yale B dataset

Figure 4.11 shows the distribution of errors for each of the algorithms on the Extended Yale B dataset. The error for SSSC was lower (0.42 instead of 0.52) than the error reported by Peng et al. (2013a) but used only 5 subspaces instead of the 38. RSSC$^{\text{no}}$ had the best error ($\mu = 0.429, Q_2 = 0.438$), and mutual info ($\mu = 0.413, Q_2 = 0.410$).

There were significant differences ($p < 0.000$ for both measurements) between the distributions. Pairwise testing showed that the difference between RSSC$^{\text{no}}$ and RSSC$^{\text{rep}}$ were significant ($p < 0.000$) but there was no significant difference between RSSC$^{\text{no}}$ and SSSC. The results for all algorithms were significantly worse ($p < 0.000$) than SSC for the clustering error $\mu = 0.075, Q_2 = 0.069$) and mutual info ($\mu = 0.854, Q_2 = 0.867$).

4.4.3 Hopkins 155 dataset

Figure 4.12 shows the distribution of errors for each of the algorithms on the Hopkins 155 datasets. No significant differences were found between the error distributions (for both clustering error and mutual information $p > 0.3$). Compared to SSC the results were significantly worse ($p < 0.000$) on both the clustering error ($\mu = 0.022, Q_2 = 0.000$) and mutual info ($\mu = 0.934, Q_2 = 1.000$.

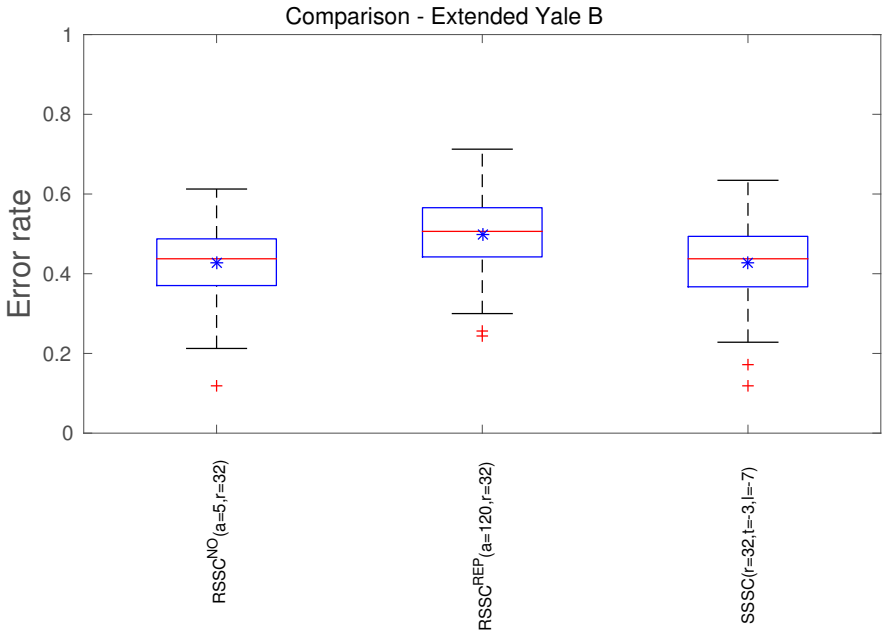

Figure 4.11: Distribution of errors on Extended Yale B dataset.

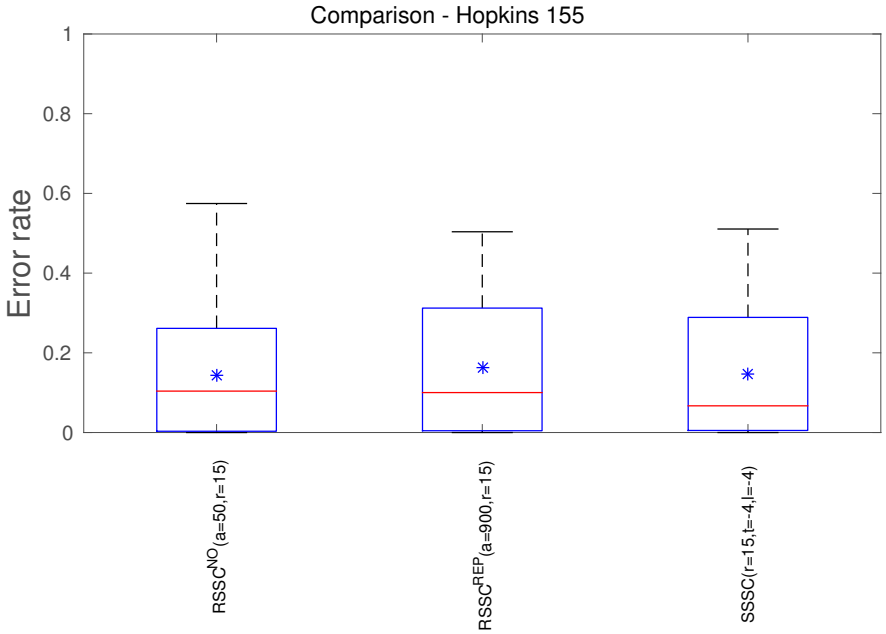

Figure 4.12: Distribution of errors on the Hopkins 155 dataset.

Chapter 5

Discussion

5.1 Comparing sparse modeling representatives selection to hierarchical sparse representatives

The theoretical results that are derived in Chapter 3 show that SMRS and HSR have approximately the same results. Approximately, because the equivalence is true for convex hulls and SMRS returns an approximation of the convex hull. Furthermore, the complexity analysis shows that when the number of found representatives is not high (less than 75% of the dataset), HSR is faster than SMRS using only two subsets. Using a higher branching factor or finding less representatives makes HSR even faster. Consequently, it is very advantageous to use HSR instead of SMRS when the dataset has many data points.

The empirical results on the precision and accuracy support these theoretical results. For all amount of (non-)representatives there is a big overlap (accuracy) between the representatives of SMRS and HSR. Furthermore, the HSR representatives are a good approximation (precision) of SMRS representatives.

The amount of representatives and the noise level of the dataset have an influence on the precision. With small numbers of representatives the precision is lower than with larger amounts of representatives. Without noise the precision is much higher than when noise is added. An explanation is that in datasets without noise there is a much clearer distinction between data points that are on the convex hull and which do not because only the lower dimensional manifold matters.

The correlation between the normalized indexes of the representatives shows a similar result. With small in-sample sizes the correlation is small and when the in-sample size increases the correlation also increases. Again there is a difference between the generated linear subsets with and without noise. The correlation of

the dataset without noise is much higher than the dataset with noise when the in-sample size is small. Without noise the representatives algorithms can more easily find the dimensions of each subspace that matter.

The difference between the normalized index of SMRS and HSR has a non-zero mean for most tests. This can be explained by the cut-off of the measurements at zero and one. Especially for normalized indexes of SMRS that are close to these borders the error could be distorted. A linear regression with the normalized indexes of SMRS as a regressor and the normalized indexes of HSR as a dependent variable would shed more light onto the amount of the data that is explained by an intercept and the regressor.

Summarizing the results, when the noise is not too high the representatives of HSR are comparable to SMRS. When the number of representatives increases, the accuracy and precision naturally become larger. In the limit all representatives are included by both algorithms and the accuracy and precision becomes one. But when the number of representatives increases the correlation also increases and therefore the ordering of the representatives is also similar.

5.2 Clustering with sparse modeling representatives selection

The clustering performance was measured with the subspace clustering error and the mutual information. These two measures agreed almost always when deciding if two algorithms were significantly different. That is why only results for the subspace clustering error are shown while both results are reported.

Besides the fixed datasets, the clustering performance dependency on the cosine similarity and noise were tested. These two parameters of the generated linear subspaces were successfully modified. With these two conclusions the actual results can be discussed.

Experiment 2 obtained the optimal parameters for SSC, SSSC, RSSC$^{\text{rep}}$ and RSSC$^{\text{no}}$ for the tree different datasets. The parameters for SSC and SSSC clearly influence the performance. However, the different α values for RSSC$^{\text{rep}}$ and RSSC$^{\text{no}}$ only differentiated the results slightly. For the generated linear subspaces there was no clear significant pattern in the performance of different α values. The same is true for the Extended Yale B dataset. However, for the Hopkins 155 datasets it was beneficial to select very few representatives (low α), very many representatives (high α) or very many non-representatives (low α). Despite these positive results on the Hopkins 155 dataset, no clear relation between the α parameter and the performance of both the RSSC$^{\text{rep}}$ and RSSC$^{\text{no}}$ algorithm is found.

Likewise, the direct comparison between RSSC with and without representa-

tives on one hand and SSSC on the other hand shows that there is no improvement by using (non-)representatives instead of random data points as in-sample set in SSC. Also, there are no clear differences between using representatives or non-representatives. Only on the Extended Yale B dataset the RSSC^{no} performs slightly better than the RSSC^{rep}. Again, we must conclude that there is apparently no large difference between representatives, non-representatives and random data points when choosing in-sample sets for SSSC.

These results are unexpected because random sampling might be good for large datasets with large number of features, the datasets in this research have a clear lower-dimensional underlying structure. Furthermore, the used in-sample size for all datasets was just above or just below the theoretical number of samples needed. For this reason, smartly chosen data points for the in-sample should be beneficial.

But, however, this did not happen. One explanation might be that the variance of the latter dimensions of the lower-dimensional subspace are not that important. Consequently, using only very few data points from a subspace already explains much of the variance in the data points. Looking at the SVD of the Extended Yale B dataset and Hopkins 155 dataset (Elhamifar and Vidal, 2013) and the SVD of the generated linear subspaces (Figure C.2) it can be concluded that indeed much of the variance is indeed explained by only a few dimensions.

Another explanation is that both the representatives and non-representatives have advantageous and disadvantageous in different stages of the SSSC algorithm. As explained in Section 2.1.2 the representatives are theoretically perfect to represent all the data points and should thus make the out-of-sample part of SSSC much easier. However, clustering the representatives is much harder because by definition those representatives are hard to represent as a linear combination of each other. Using non-representatives turns the advantage into a disadvantage and vice-versa.

As a last attempt to detect the differences between RSSC and SSSC the relation between the cosine angle and in-sample size was investigated. There were only two significant differences and those could be explained well by chance. However, the results could also suggest that RSSC is a tiny bit better in clustering subspaces with an in-sample size that is too little or just enough. But more surprisingly, the datasets with low cosine similarities and a too small in-sample sizes to represent all variation in the subspace were clustered perfectly. This suggests that only a few data points already reconstructs the subspace well enough to distinguish it from the other subspaces. Consequently, to test the hypothesis that RSSC might be better with small in-sample size, even smaller in-sample size should be used.

5.3 Future research

More research into the parameters that govern HSR is needed. For larger datasets, h definitely needs to be bigger than two and consequently the effect of the branching factor h on the empirical similarity between SMRS and HSR should be known.

Using a smart initialization for the in-sample set of SSSC did not improve the clustering performance. As proposed earlier this might be explained by the fact that the variance in the data points is explained much by only a few singular vectors. Testing this hypothesis thus involves clustering a dataset with only a tiny in-sample set. Furthermore, in this research we added noise to the generated linear subspaces but actually Elhamifar and Vidal (2013) used the smallest singular vector of the subspace as a part of their prove. Manipulating this smallest singular vector might than reveal the small region of datasets where using SMRS or HSR as an initialization of SSSC improves the clustering.

5.4 Conclusion

A hierarchical application of SMRS called HSR was tested both theoretically and empirically for equivalence with SMRS. HSR reduces the complexity of SMRS under mild circumstances while returning similar representatives. Consequently, HSR can be used to find representatives for large datasets, both in the number of features and the number of data points.

However, representatives or non-representatives from SMRS do not improve the performance of SSSC, a clustering algorithm for large datasets. Consequently, SSSC remains the algorithm that should be used for fast clustering of large datasets. However, these fast obtained results are not always as satisfactory as those obtained with SSC and thus the search for a fast high-dimensional data clustering algorithm continues.

Appendix A

The curse of dimensionality

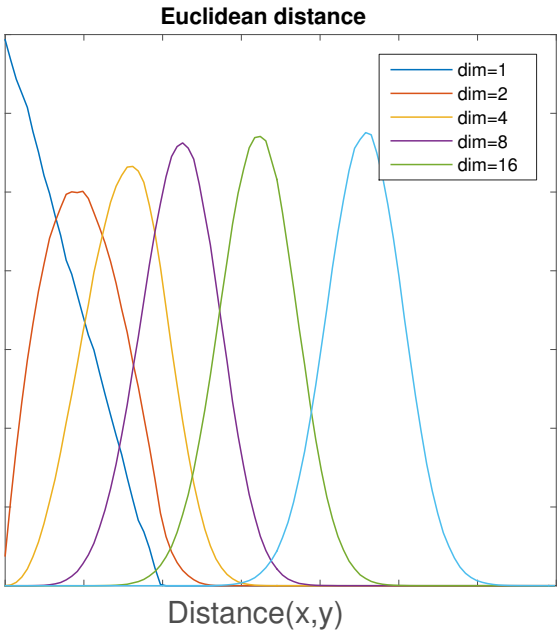

Figure A.1: Histogram of Euclidean distances between randomly sampled data points with different number of dimensions

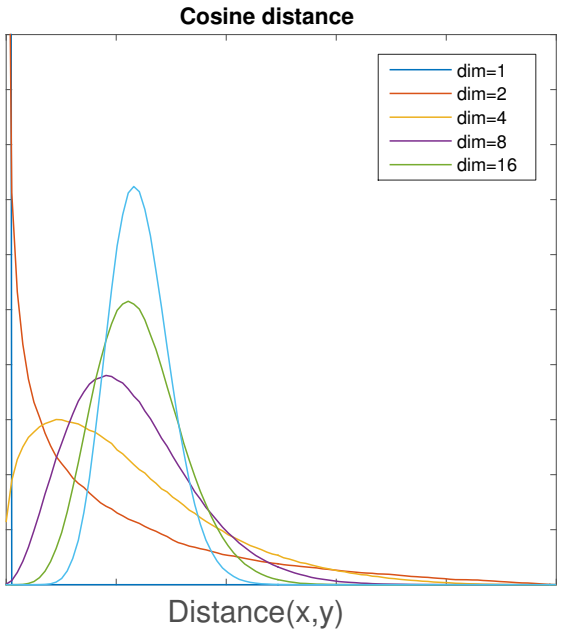

Figure A.2: Histogram of cosine distances between randomly sampled data points with different number of dimensions

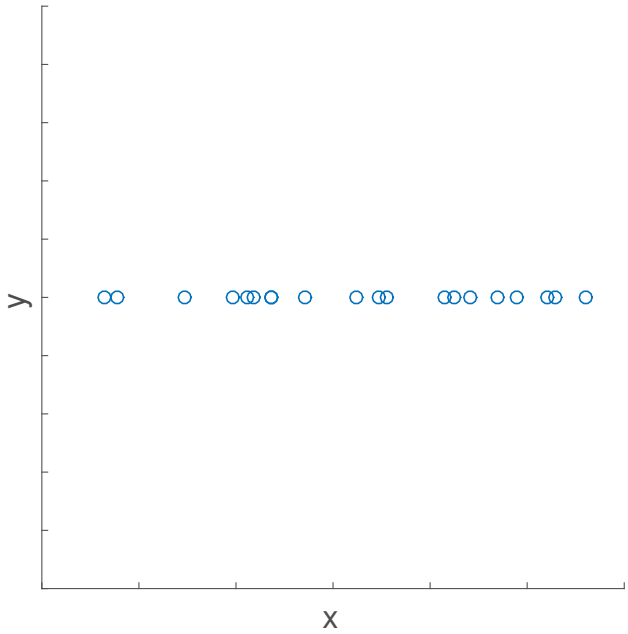

Figure A.3: Twenty data points randomly distributed in a 1-dimensional dataspace

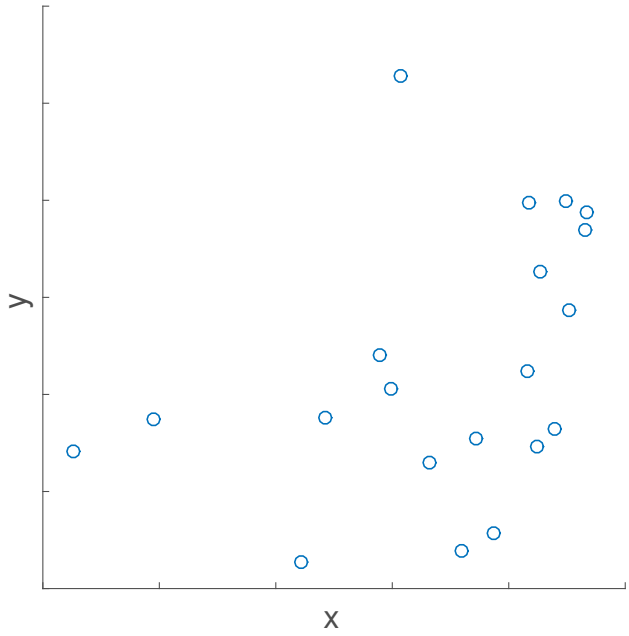

Figure A.4: Twenty data points randomly distributed in a 2-dimensional dataspace

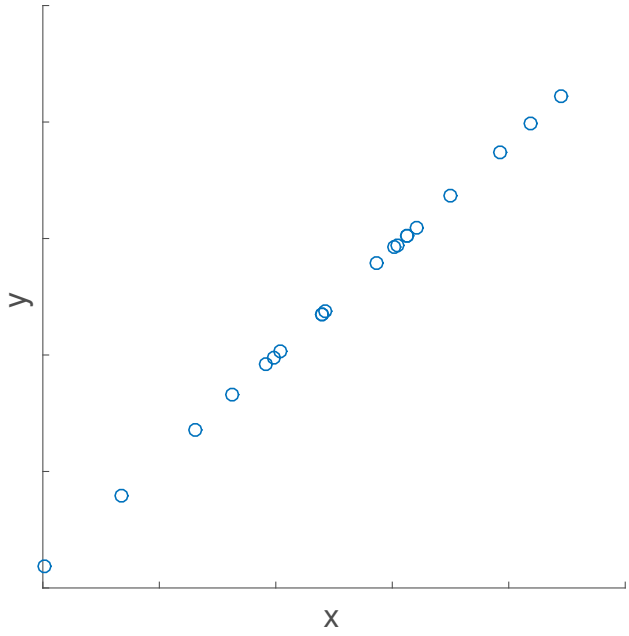

Figure A.5: One-dimensional data in a two-dimensional dataspace

Appendix B

Parameter optimization results

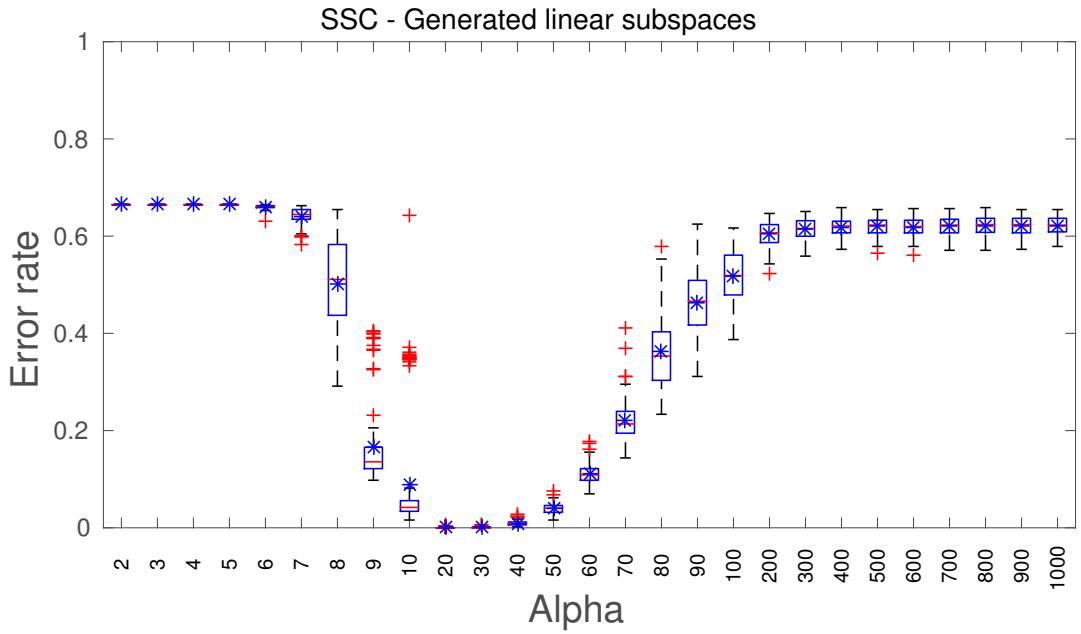

Figure B.1: Subspace clustering error for SSC with different values of alpha for generated linear subspaces.

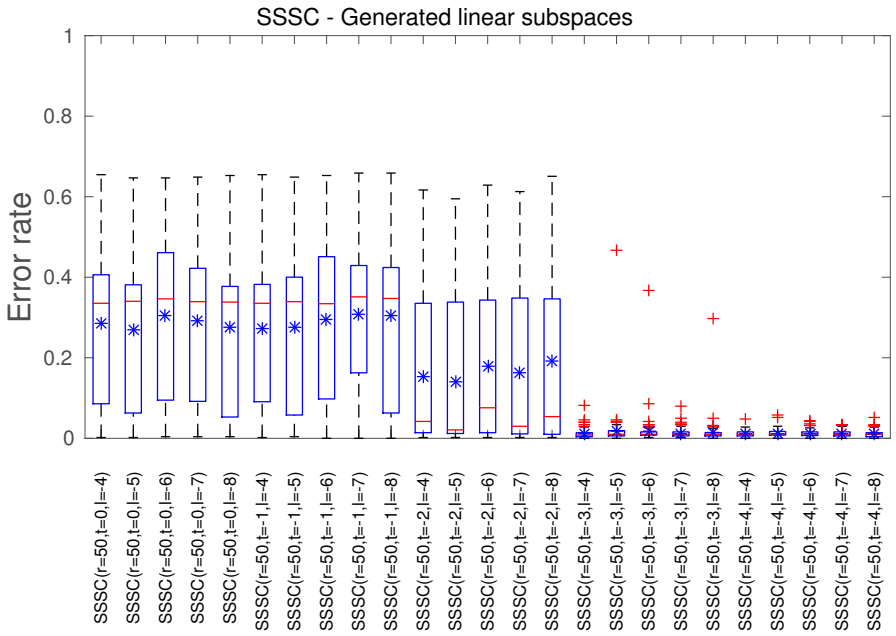

Figure B.2: Subspace clustering error for SSSC with different values for λ and tolerance for generated linear subspaces.

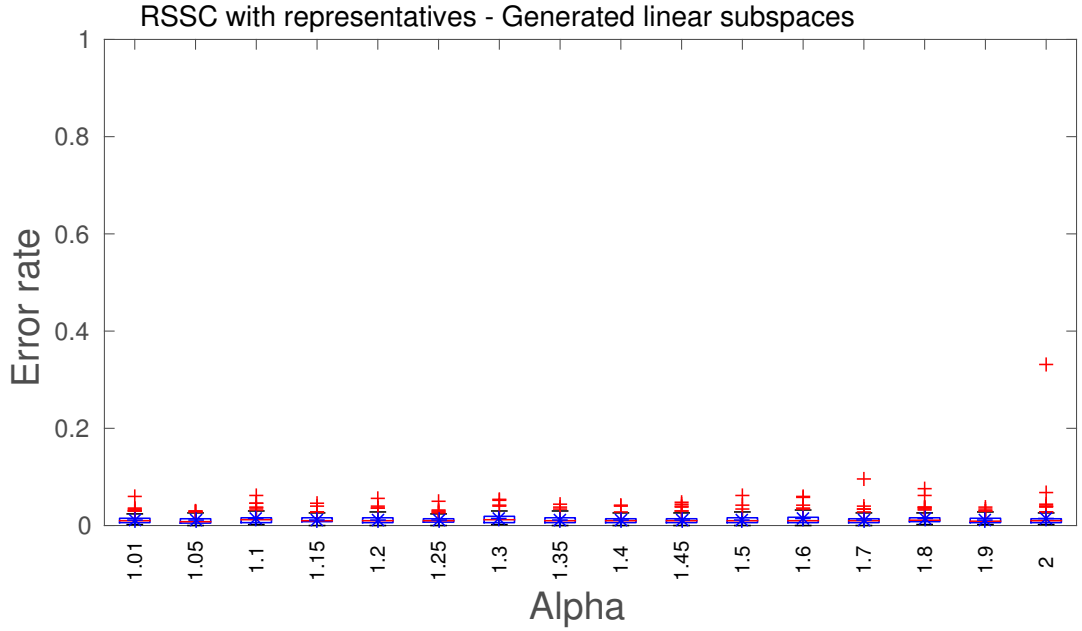

Figure B.3: Subspace clustering error for RSSC$^{\text{rep}}$ with different values of alpha for generated linear subspaces.

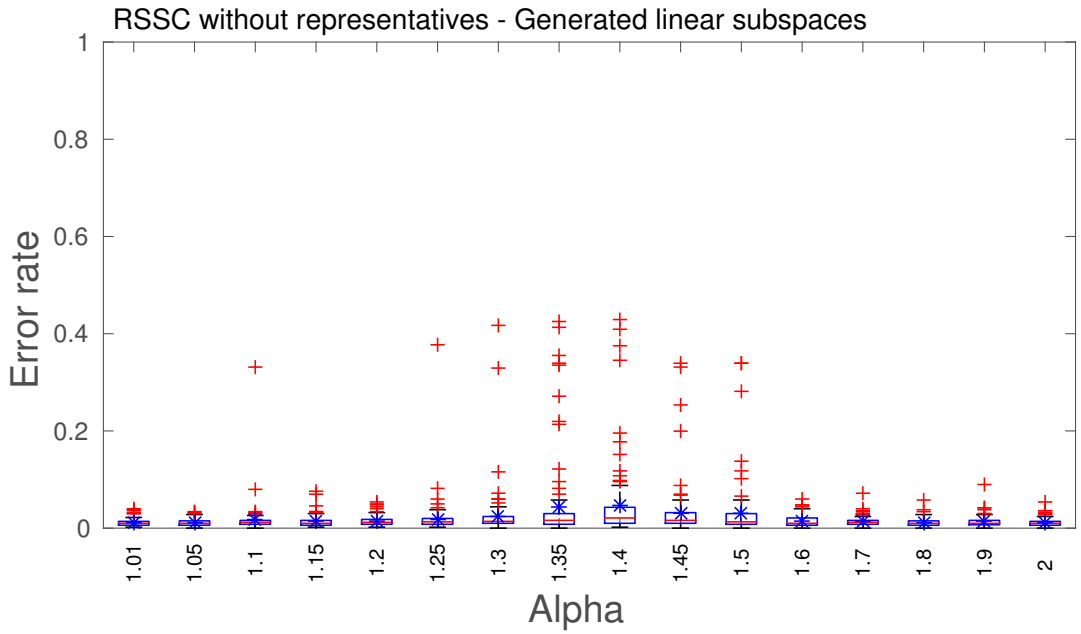

Figure B.4: Subspace clustering error for RSSC^{no} with different values of alpha for generated linear subspaces.

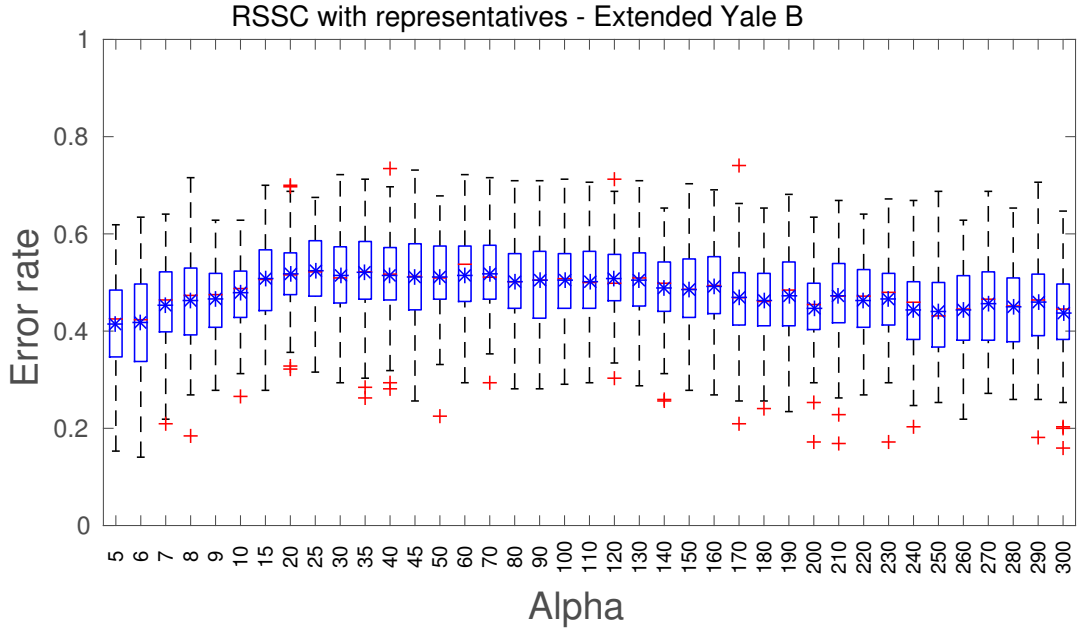

Figure B.5: Subspace clustering error for RSSC^{rep} with different values of alpha for the Extended Yale B dataset.

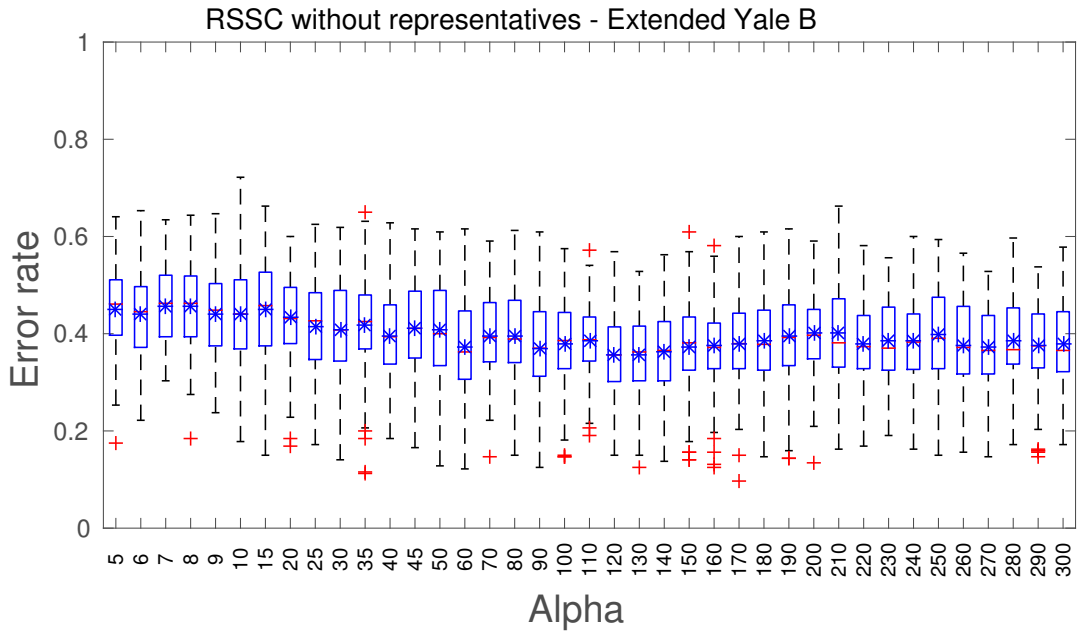

Figure B.6: Subspace clustering error for RSSC^{no} with different values for alpha for the Extended Yale B dataset.

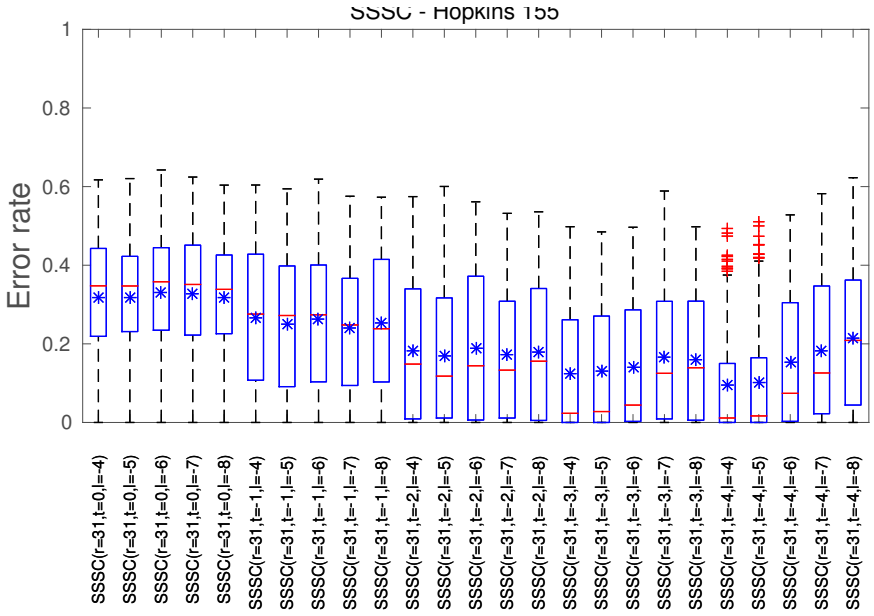

Figure B.7: Subspace clustering error for SSSC with different values for λ and tolerance for the Hopkins 155 dataset.

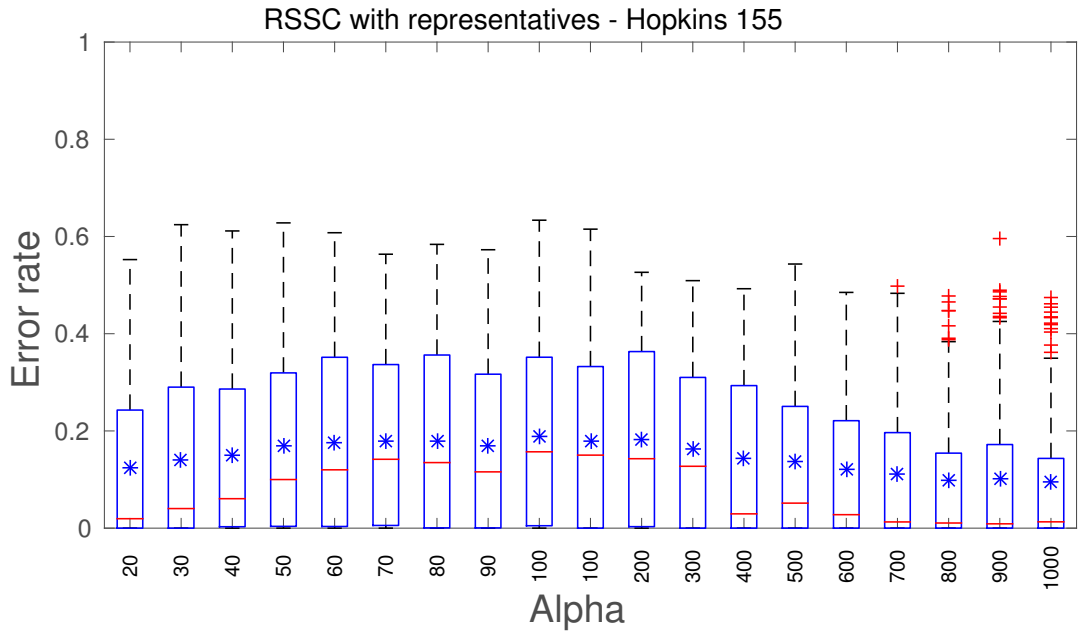

Figure B.8: Subspace clustering error for RSSC$^{\text{rep}}$ with different values for α for the Hopkins 155 dataset.

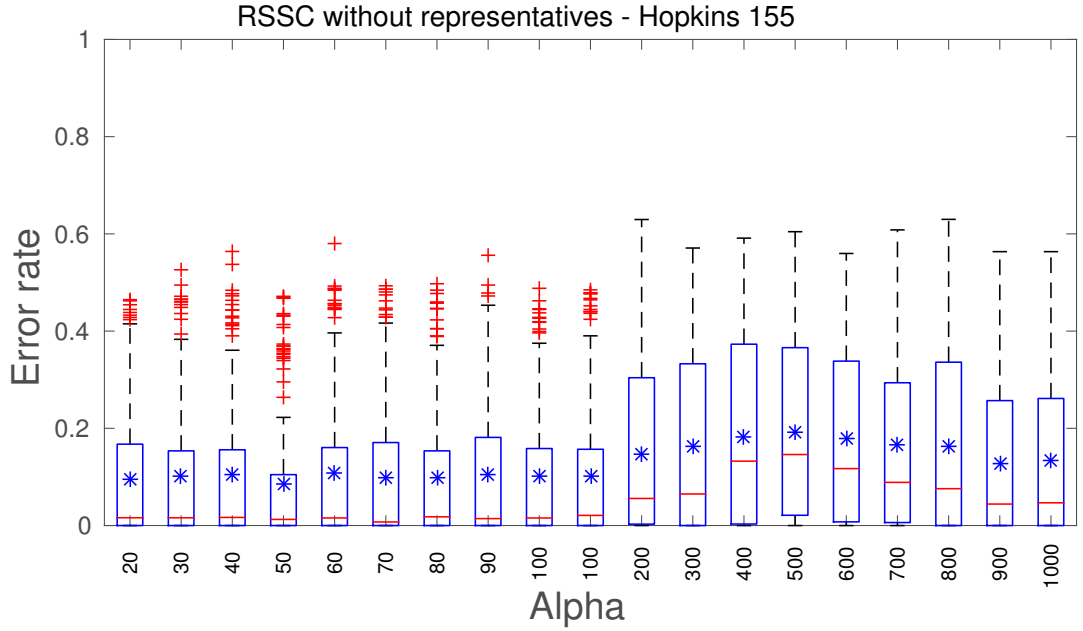

Figure B.9: Subspace clustering error for RSSC$^{\text{no}}$ with different values for α for the Hopkins 155 dataset.

Appendix C

Additional results

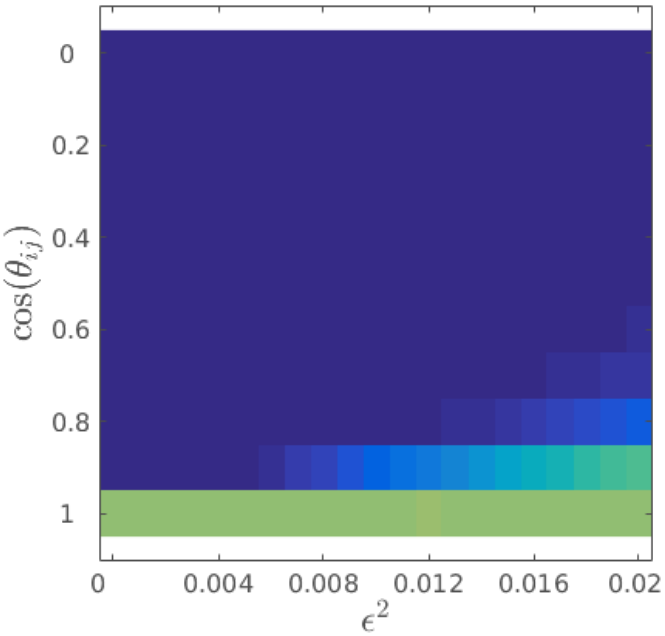

Figure C.1: Clustering error for different levels of noise ϵ^2 and cosine similarity between the subspaces. Clustering is not succesfull when either the noise or the cosine similarity is too high. Generated with $N = 500, D = 2000, d = 10, n = 3$.

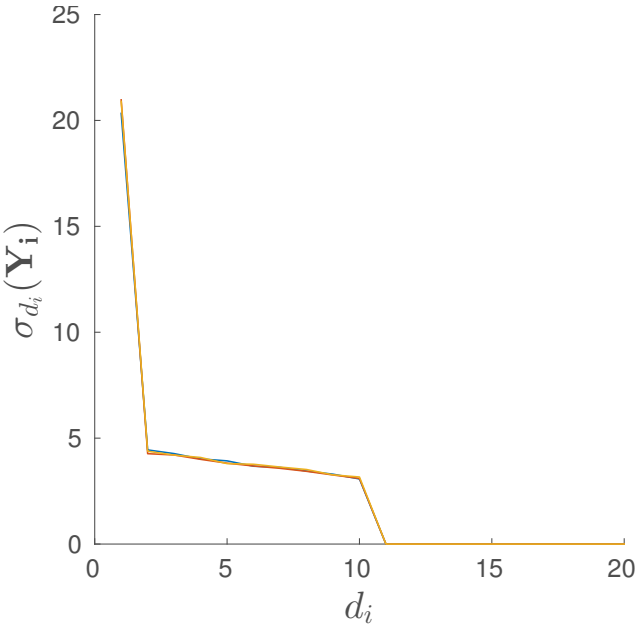

Figure C.2: The singular values of the three subspaces of a generated linear subspace with 501 datapoints in 2000 dimensions with three underlying subspaces of ten dimensions. There is no noise added to the dataset.

Bibliography

Aggarwal, C. C., Hinneburg, A., and Keim, D. A. (2001). On the surprising behavior of distance metrics in high dimensional space. In *Proceedings of the 8th International Conference on Database Theory*, pages 420–434, Berlin. Springer.

Aggarwal, C. C. and Reddy, C. K. (2013). *Data clustering: algorithms and applications*. CRC Press.

Barber, C. B., Dobkin, D. P., and Huhdanpaa, H. (1996). The quickhull algorithm for convex hulls. *ACM Transactions on Mathematical Software*, 22(4):469–483.

Basri, R. and Jacobs, D. (2003). Lambertian reflectances and linear subspaces. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 25(2):383–390.

Bishop, C. M. (2006). *Pattern Recognition and Machine Learning*. Springer.

Cai, D., He, X., and Han, J. (2005). Document clustering using locality preserving indexing. *IEEE Transactions on Knowledge and Data Engineering*, 17(12):1624–1637.

Chen, J. and Yang, J. (2014). Robust subspace segmentation via low-rank representation. *IEEE Transactions on Cybernetics*, 44(8):1432–1445.

Costeira, J. and Kanade, T. (1998). A multibody factorization method for independently moving objects. *International Journal of Computer Vision*, 29(3):159–179.

de Berg, M., Cheong, O., van Kreveld, M., and Overmars, M. (2008). *Computational Geometry*. Springer, third edition.

Derksen, H., Ma, Y., Hong, W., and Wright, J. (2007). Segmentation of multivariate mixed data via lossy coding and compression. In Chen, C. W., Schonfeld, D., and Luo, J., editors, *IEEE Transactions on Pattern Analysis and Machine Intelligence (TPAMI)*, volume 6508, pages 65080H–65080H–12.

Donoho, D. L. (2006). For most large underdetermined systems of equations, the minimal L1-norm near-solution approximates the sparsest near-solution. *Communications on Pure and Applied Mathematics*, 59(7):907–934.

Elhamifar, E., Sapiro, G., and Vidal, R. (2012). See all by looking at a few: Sparse modeling for finding representative objects. In *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, pages 1600–1607.

Elhamifar, E. and Vidal, R. (2009). Sparse subspace clustering. In *IEEE Conference on Computer Vision and Pattern Recognition*, pages 2790–2797. IEEE.

Elhamifar, E. and Vidal, R. (2010). Clustering disjoint subspaces via sparse representation. In *IEEE International Conference on Acoustics Speech and Signal Processing*, pages 1926–1929.

Elhamifar, E. and Vidal, R. (2013). Sparse subspace clustering: algorithm, theory, and applications. *IEEE transactions on pattern analysis and machine intelligence*, 35(11):2765–81.

Fischler, M. a. and Bolles, R. C. (1981). Random sample consensus: a paradigm for model fitting with applications to image analysis and automated cartography. *Communications of the ACM*, 24(6):381–395.

Georghiades, A. S., Belhumeur, P. N., and Kriegman, D. J. (2001). From few to many: Illumination cone models for face recognition under variable lighting and pose. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 23(6):643–660.

Kriegel, H.-P., Kröger, P., and Zimek, A. (2009). Clustering high-dimensional data: A survey on subspace clustering, pattern-based clustering, and correlation clustering. *ACM Trans. Knowl. Discov. Data*, 3(1):1–58.

Kuhn, H. W. (1955). The Hungarian method for the assignment problem. *Naval Research Logistics Quarterly*, 2:83–97.

Lay, D. C. (2012). *Linear Algebra and its application - 4th Edition*.

Lee, K. C., Ho, J., and Kriegman, D. J. (2005). Acquiring linear subspaces for face recognition under variable lighting. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 27(5):684–698.

Luxburg, U. (2007). A tutorial on spectral clustering. *Statistics and Computing*, 17(4):395–416.

Mustafa, N. H. (2004). k-Means Projective Clustering. In *Proceedings of the twenty-third ACM SIGMOD-SIGACT-SIGART symposium on Principles of database systems*.

Osborne, M. R., Presnell, B., and Turlach, B. a. (2000). A new approach to variable selection in least squares problems. *IMA Journal of Numerical Analysis*, 20(3):389–403.

Peng, X., Zhang, L., and Yi, Z. (2013a). Inductive sparse subspace clustering. *Electronics Letters*.

Peng, X., Zhang, L., and Yi, Z. (2013b). Scalable sparse subspace clustering. *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, pages 430–437.

Tomasi, C. and Kanade, T. (1992). Shape and motion from image streams under orthography: a factorization method. *International Journal of Computer Vision*, 9(2):137–154.

Tron, R. and Vidal, R. (2007). A benchmark for the comparison of 3-d motion segmentation algorithms. In *IEEE Conference on Computer Vision and Pattern Recognition*, pages 1–8.

Vidal, R. (2011). A tutorial on subspace clustering. *IEEE Signal Processing Magazine*, 28(2):52–68.

Vidal, R., Ma, Y., and Sastry, S. (2005). Generalized principal component analysis (GPCA). *IEEE transactions on pattern analysis and machine intelligence*, 27(12):1945–59.

Yang, A., Sastry, S., Ganesh, A., and Ma, Y. (2010). Fast 1-minimization algorithms and an application in robust face recognition: A review. *Image Processing (ICIP), ...*, (6):1–4.